Optimizing Earthmoving Operations Using Computer Simulation

Mohamed Marzouk

A Thesis in the

Department of Building, Civil and Environmental Engineering

Presented in Partial Fulfillment of the Requirements

For the Degree of Doctor of Philosophy (Building Studies) at

Concordia University

Montreal, Quebec, Canada

February 2002

ABSTRACT

OPTIMIZING EARTHMOVING OPERATIONS USING COMPUTER SIMULATION

Mohamed Marzouk, Ph.D.
Concordia University, 2002

This thesis presents a new methodology for optimizing earthmoving operations using computer simulation and genetic algorithms. It provides an optimization tool geared towards selection of near-optimum fleet configurations. The optimization used in the selection of fleets accounts for availability of equipment and aims at minimizing the total project cost or duration. The simulation process, in the proposed methodology, utilizes discrete event simulation (DEVS) and object oriented modeling. Different features of object orientation are employed including classes, dynamic data structure and polymorphism. The three-phase simulation approach, rather than process interaction, was employed to control the dynamics of the simulation process and track involved activities. This simulation approach is considered most appropriate for object oriented simulation (OOS). The optimization process uses a developed genetic algorithm to search for a near-optimum fleet configuration that reduces project total cost. The algorithm considers a set of qualitative and quantitative variables that influence the production of earthmoving operations. Qualitative variables represent the models of equipment used in each fleet scenario, whereas, quantitative variables represent the number of equipment involved in each scenario.

The proposed methodology accounts for: 1) uncertainties associated with earthmoving operations; 2) optimization of project duration or its total cost, considering equipment availability; and 3) realistic estimates of haulers' travel time. It also makes full use of object oriented features and is implemented in a prototype software system named *SimEarth*. The system consists of five main components: 1) EarthMoving Simulation Program (*EMSP*); 2) Equipment Cost Application (*ECA*); 3) Equipment Database Application (*EDA*); 4) EarthMoving Genetic Algorithm (*EM_GA*), and 5) Output Reporting Module (*ORM*). Beside these main components, *SimEarth* is supported by: a) Hauler's Travel Time Application (*HTTA*), and b) EarthMoving Markup Application (*EMMA*). All system components are implemented in Microsoft (*MS*) environment except the dynamic sub-module of *ORM* component, which is implemented utilizing *"Proof Animation"* software. Five numerical examples were analyzed in order to validate and demonstrate the essential features of the system's components. A comprehensive case study of an actual project was analyzed in order to test the performance of the developed system (including the dynamic interaction among its components) and to illustrate the practical features of the developed methodology. The project involves the construction of a large rockfill dam, located in the northern part of the province of Quebec.

ACKNOWLEDGEMENTS

Humble thanks are first offered to God

I wish to express my profound gratitude and appreciation to my supervisor Dr. Osama Moselhi for his time, guidance and academic support during my research.

My thanks also to the faculty, staff and my colleagues at the Department of Civil, Building and Environmental Engineering for their support and encouragement. The valuable time and the efforts dedicated from Mrs. Juliet Dunphy and Mr. Sylvain Belanger are appreciated.

Finally, my special thanks to my father, mother and fiancée for their continuous support and encouragement during the course of this research.

The financial support provided by J.W. McConnell Memorial Graduate Fellowship and the National Sciences and Engineering Council of Canada grant is greatly acknowledged.

TABLE OF CONTENTS

CHAPTER 3

CHAPTER 4

CHAPTER 5

CHAPTER 6

CHAPTER 8

LIST OF FIGURES

CHAPTER 1

CHAPTER 2

CHAPTER 3

CHAPTER 4

CHAPTER 5

CHAPTER 6

CHAPTER 7

APPENDIX C

LIST OF TABLES

CHAPTER 2

CHAPTER 3

CHAPTER 5

CHAPTER 6

CHAPTER 7

APPENDIX C

CHAPTER 1

INTRODUCTION

1.1 General

Earthmoving operations are commonly encountered in heavy civil engineering projects. In general, it is rare to find a construction project free of these operations. For large-scale projects (e.g. dams, highways, airports and mining), earthmoving operations are essential, frequently representing a sizable scope of work. Considerable efforts have been made in the development of efficient methods and systems for estimating earthmoving production, recommending appropriate fleets of equipment, and optimizing resources used in this class of projects (Carmichael 1986-a, Mayer and Stark 1981, Touran 1990, Smith et al 1995, Shi 1995, Chao and Skibniewski 1994, FPC 1998, Oloufa 1993, Martinez 1998, AbouRizk and Hajjar 1998, McCabe 1998, Smith et al 2000).

Earthmoving operations are frequently performed under different job conditions, which may give rise to uncertainty. This includes equipment breakdown, inclement weather and unexpected site conditions (Moselhi and Marzouk 2000). Traditional scheduling using network techniques (critical path method (CPM), precedence diagram method (PDM)) and line of balance (LOB) does not directly taking into account such uncertainties (Sawhney and AbouRizk 1995). Also, earthmoving operations have a cyclic nature which can ideally be represented by simulation (Touran 1990). In fact, it is essential, in modeling these operations,

to consider the dynamic interaction among the individual pieces of equipment in a production fleet. Figure 1.1 illustrates a snapshot for equipment involved in an earthmoving operation; a truck is being loaded by load equipment, three are waiting in their queue and one is about to finish its return trip. Therefore, modeling these operations requires a consideration for such interaction.

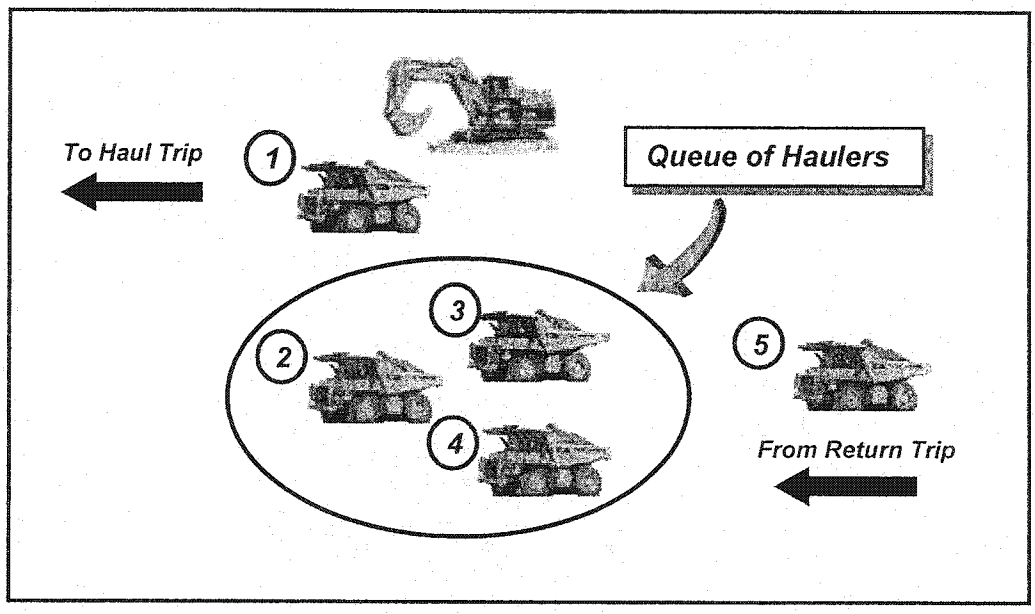


Figure 1.1 Equipment Interaction

1.2 Computer Simulation

Simulation is one of the techniques that has been used to model uncertainties involved in construction operations (AbouRizk and Hajjar 1998). Although simulation is a powerful tool for modeling construction operations, the application of simulation is still limited in the construction domain. This has generally been

attributed to the difficulty in learning and applying simulation languages to industry (Sawhney and AbouRizk 1996, Oloufa et al 1998, Touran 1990).

The simulation process is an iterative process which involves different steps. It has been defined as "imitation of a real-world process or system over time" (Banks et al 2000). Modeling construction operations utilizing discrete event simulation, requires the modeler to define three main elements (Schriber and Brunner 1999): project, experiments and replications (see Figure 1.2). A "*project*" is performed to study a certain operation which has specific characteristics, for example, an earthmoving operation that contains a definite scope of work and specific road characterisitics. An "*experiment*" represents one alternative of the project under consideration by changing the resources assigned for the execution of the project and/or its individual activities. A "*replication*" represents one execution of an experiment within the project. It is assumed, in the proposed methodology, that all projects' experiments have the same number of replications.

Modeling utilizing simulation can be applied either in a general or special purpose simulation environment. General purpose simulation (GPS) is based on formulating a simulation model for the system under study, running the simulation and analyzing the results in order to decide whether the system is acceptable or not. In case of being unacceptable, the process is re-iterated and a new alternative system is considered.

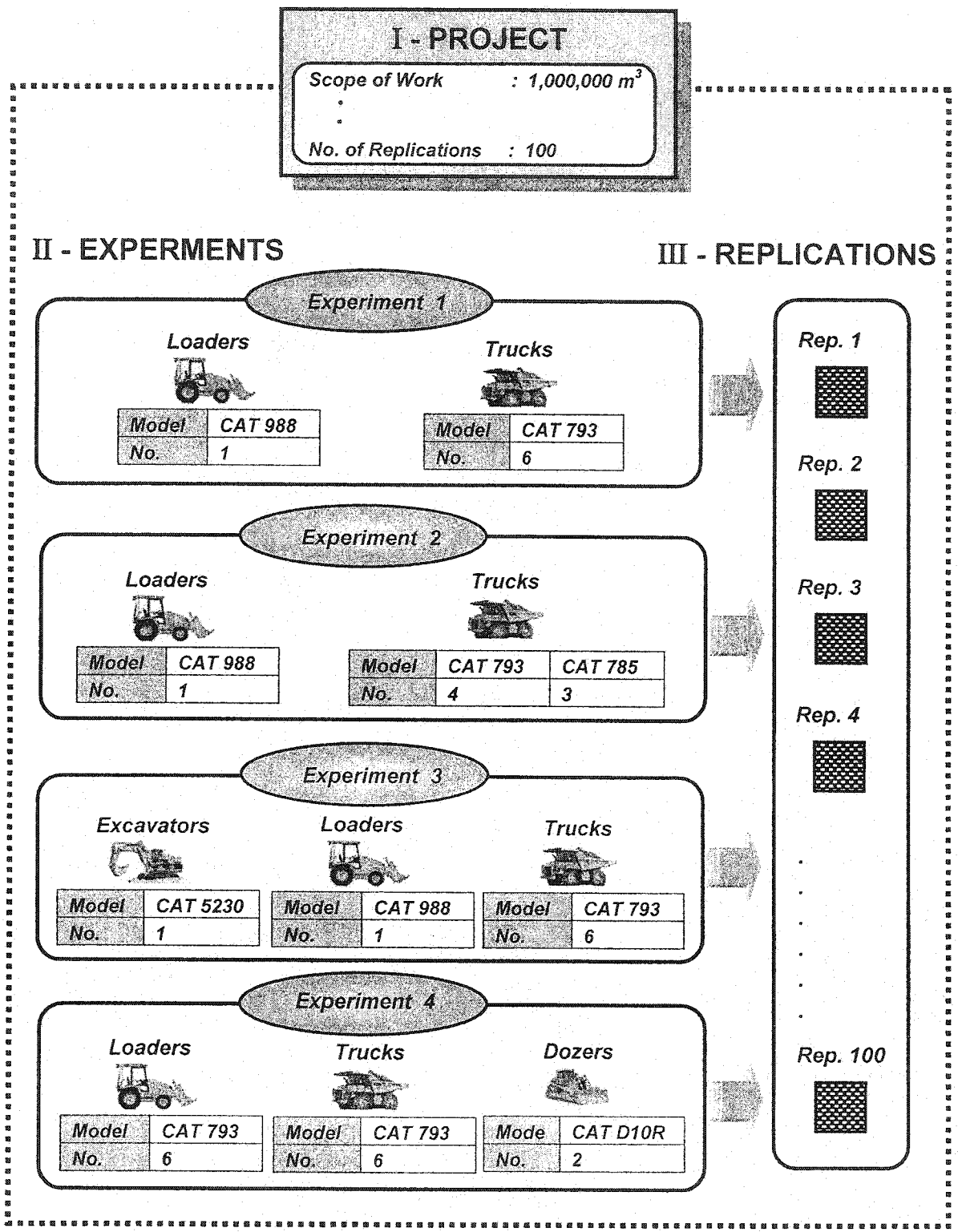


Figure 1.2 Elements of Discrete Event Simulation

Different GPS software systems have been developed for a wide range of industries: AweSim (Pritsker et al 1997) and GPSS/H (Crain 1997); for construction: Micro-CYCLONE (Halpin and Riggs 1992) and STROBOSCOPE (Martinez 1996).

Special purpose simulation (SPS) is based on creation of a platform or a template for a specific domain of application (Marzouk and Moselhi 2000, AbouRizk and Hajjar 1998). The steps for simulation, in this case, are the same as in the GPS case except for the first step (construct simulation model) since the platform has the characteristics and behavior of the system under study. Also, the modification is limited to the input parameter(s) of a pre-defined system and not to the characteristics and behavior of the system.

1.3 Research Objectives

The objective of this research is to develop an efficient and practical methodology for selecting near-optimum equipment fleets used in earthmoving operations. The optimization used in the selection of fleets accounts for availability of equipment to contractors and aims at minimizing the total cost or duration of the project. To achieve the stated objective, the following sub-objectives are carried out:

1) Capture and model the interaction among earthmoving equipment utilizing computer simulation;

2) Develop an algorithm that provides near-optimum fleet configurations which

satisfy minimum duration or minimum project total cost;

3) Develop an automated environment for storing and processing data pertaining to the characteristics of earthmoving equipment and their cost components;

4) Develop a model that provides a realistic estimates of haulers' travel time; and

5) Implement the developed methodology in a prototype earthmoving simulation system that serves as a proof of concept for the developed methodology.

1.4 Organization of the thesis

Chapter 2 presents a literature review of simulation modeling techniques in general and those focusing primarily on construction applications. Included also in this chapter is a review of the methods used to model earthmoving operations.

Chapter 3 provides an overview of the proposed methodology including the layout of the system and the design of the simulation engine. In addition, a numerical example is presented to test the performance of the engine.

The developed equipment database and equipment cost application are described in Chapter 4 along with a numerical example towards the end of the chapter.

In Chapter 5, a fuzzy clustering model for estimating haulers' travel time is

presented. A numerical example is then presented to validate this model. Included also in this chapter is a multi-attribute decision support model for estimating projects' markup.

Chapter 6 presents a methodology for simulation optimization utilizing Genetic Algorithms (GAs). The characteristics of the developed algorithm will be presented. A numerical example is presented towards the end of the chapter.

Chapter 7 presents the implementation of the developed system. It illustrates its specially designed user interfaces and the customized output reports. A comprehensive case study, of an actual project, is presented to demonstrate the process of fleet selection and application of the developed methodology and to illustrate its practical features. The project involves the construction of a large rockfill dam, located in the northern part of the province of Quebec.

In Chapter 8, the contributions of the research proposed in this thesis are summarized along with an outline of future work.

CHAPTER 2

LITERATURE REVIEW

2.1 General

Simulation has been applied in different fields including computer systems, manufacturing, business, environmental, and construction (Roberts and Dessouky 1998, Banks et al 2000). Shannon (1992) defined the simulation as "the process of designing a model of a real system and conducting experiments with this model for the purpose of either understanding the behavior of the system and/or evaluating various strategies for the operation of the system". Another definition for computer simulation by Pritsker et al (1997) as "the process of designing a mathematical-logical model of a real system and experimenting with the model on a computer". Maria (1997) defined the mathematical models that are utilized in simulation to stochastic (probabilistic input(s) and output(s)) and dynamic (time-varying) rather than deterministic and static.

Modeling techniques using simulation are categorized according to the changing nature of the dependent variable(s) (Pritsker et al 1997). This change can be discrete, continuous or combined over the entire simulation time. *Discrete Simulation* takes place when the change of the dependent variables occurs at specific time points which represent event times (see Figure 2.1-a). *Continuous Simulation* takes place when the change of the dependent variables occurs continuously over the time of the process being modeled (see Figure 2.1-

b). Usually, equations are defined to represent the dynamic nature of the dependent variables in continuous simulation. *Combined Simulation* takes place when the change of the dependent variables occurs discretely, continuously or continuously with a set of discrete jumps over the time (see Figure 2.1-c). It should be noted that the change of the dependent variables in construction operations occurs discretely. For example, in earthmoving operations, the decrease and increase of earth at loading and dumping areas occur at discrete time points. Therefore, the discrete event simulation (DEVS) technique is utilized in the development of the proposed research methodology.

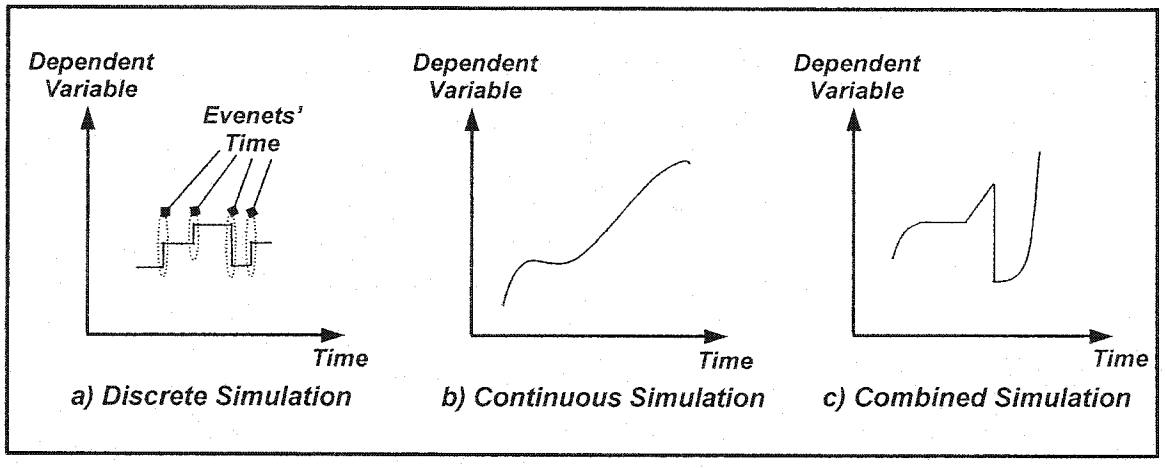


Figure 2.1 Simulation Modeling Techniques (Pritsker et al 1997)

Shannon (1992) listed seven main purposes of simulation: 1) evaluation of a proposed system; 2) comparison between alternative proposals; 3) prediction of a system performance under different conditions; 4) sensitivity analysis to determine the most significant factors affecting the performance of a system; 5) optimization in order to determine the best overall response of a system; 6) functional relations to recognize any relationship among the system's significant

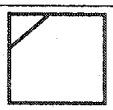
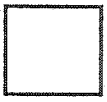
factors; and 7) bottlenecks analysis to identify the factors that cause system delays. In the following sections, a review of the computer simulation applications in construction is presented. All modeling techniques that have been applied to earthmoving operations are also briefly described.

2.2 Application of Simulation in Construction

Considerable efforts have been made to model construction operations utilizing simulation. These include Halpin (1977), Paulson (1978), Ioannou (1989), Oloufa (1993), Tommelein et al (1994), Sawhney and abouRizk (1995), Shi (1995), Martinez (1996), McCabe (1997), Oloufa et al (1998), Hajjar and AbouRizk (1999).

CYCLONE, CYCLic Operation Network (Halpin 1977), is a modeling system that provides a quantitative way of viewing, planning, analyzing and controlling construction processes and their respective operations. CYCLONE is a network simulation language that models construction activities which have a cyclic or repetitive nature. CYCLONE consists of six basic elements (see Table 2.1): 1) *NORMAL* which represents an unconstrained work task; 2) *COMBI* which represents a constrained work task; 3) *QUEUE* which represents a waiting location for resources; 4) *FUNCTION* which describes a process function (generation or consolidation); 5) *COUNTER* which controls the iterations of the cyclic operation; and 6) *ARCS* which represent the flow logic.

Table 2.1 CYCLONE Modeling Elements (Halpin 1977)

Name	Symbol
COMBI	
NORMAL	
QUEUE	
FUNCTION	
COUNTER	
ARCS	

Different simulation implementations have been developed utilizing the CYCLONE which include INSIGHT (Paulson 1978), UM_CYLONE (Ioannou 1989), and Micro-CYCLONE (Halpin and Riggs 1992). Several construction processes and operations have been modeled utilizing CYCLONE: selecting loader-truck fleets (Farid and Koning 1994); construction of cable-stayed bridges (Abraham and Halpin 1998, Huang et al 1994); resolving construction disputes (AbouRizk and Dozzi 1993); concrete placement operations (Vanegas et al 1993); placing and finishing slab units (Hason 1994); and paving processes (Lluch and Halpin 1982, Gowda at al 1998).

Oloufa (1993) proposed an object-oriented approach for simulating construction operations. In this approach, the system at hand is modeled by creating objects

of classes which represent the system's resources and entities. These objects are interacting and communicating amongst one another via message transfer. He developed a simulation language (MODSIM) dedicated to earthmoving operations.

Tommelein et al (1994) developed an object-oriented system (CIPROS) that models construction processes by matching resource properties to those of design components. Two types of resources were distinguished in this model: product components and construction resources. Product components are design element classes which form the process being modeled. They are defined for the contractor by plans and specifications. On the other hand, construction resources are temporary equipment and material that are used during construction. In order to create a simulation model utilizing the CIPROS, the user has to go through different steps: 1) define project design and specification; 2) create activity-level plan and relate activity; 4) initialize product components; 5) identify construction resources; 6) construct simulation network (formation of the elemental simulation networks which describe the methods of construction); and finally, 7) run simulation.

HSM (Sawhney and AbouRizk 1995, Sawhney and AbouRizk 1996) is an hierarchical simulation modeling for planning construction projects which combines the concepts of work breakdown structure and process modeling. Modeling a project via HSM requires performing different stages: 1) divide the project into hierarchical structure (project, operations, and processes); 2) create

a resource library for the project (names, quantity, cost, etc.); 3) identify the sequence of operations and links (serial, parallel, cyclic, or hammock links); 4) perform process modeling utilizing the CYCLONE modeling elements along with added ones dedicated to resources and linkages of process (see Table 2.2); and finally 5) extend the model to a common level in order to run simulation.

Table 2.2 HSM Modeling Elements (Sawhney and AbouRizk 1995)

Name	Symbol	Description
ALLOCATE RESOURCE		Used to allocate resources
FREE RESOURCE		Used to free resources
PREDECESSOR		Used to define process with one or more successors processes
SUCCESSOR		Used to define process with one or more predecessors processes

RBM (Shi 1995, Shi and AbouRizk 1997) is a resource-based modeling for construction simulation which defines the operating processes into atomic models. For developing a model utilizing RBM, three steps should be performed: 1) define resource library and specifying different atomic models along with their input and output ports; 2) define project specifications including system specifications (involving r-processes "process-task level" and their connectivity, resource assignment and termination conditions); 3) model generation by

formatting different r-processes into SLAM II network statements along with linkage transition among these r-processes whether direct (where entities can be routed directly to the following r-process) or indirect (where there is a need for a conversion).

Martinez and Ioannou (1999); and Martinez (1996) developed a general purpose simulation programming language (STROBOSCOPE). To model construction operations utilizing STROBOSCOPE, the modeler needs to write a series of programming statements that defines the network modeling elements. The STROBOSCOPE has different advantages including: 1) ability to access the state of the simulation (e.g. simulation time, number of entities waiting in their queues, etc.) and 2) ability to distinguish involved resources and entities. Several construction processes and operations have been modeled utilizing STROBOSCOPE including: earthmoving (Ioannou 1999, Martinez 1998); location of temporary construction facilities (Tommelein 1999); and the impacts of changes for highway constructions (Cor and Martinez 1999).

McCabe (1997 and 1998) developed an automated modeling approach which integrates computer simulation and belief networks. Computer simulation is used to model construction operations, whereas belief networks are utilized as a diagnostic tool to evaluate project performance indices such as: 1) queue length (QL); 2) queue wait (QW); 3) server quantity (SQ); 4) server utilization (SU); and 5) customer delay (CD). These indices are calculated from the simulation statistics, subsequently the performance of the system is evaluated by the

believe network in order to reach a corrective action. This corrective action includes modifying the number and/or capacity of the involved servers or entities in the model.

Oluofa et al (1998) proposed a set of resource-based simulation libraries as an application of special purpose simulation (SPS). In this approach, simulation was performed by first selecting construction resources from the developed libraries, then linking the resources to define the logic which describes the interaction among the resources used. Different resource libraries are required to be defined in order to serve different applications. For example, the resource library for the shield tunneling (the selected implementation for their approach) contains different resources including TBM (tunnel boring machine), trains, trucks, different rail types, vertical conveyor, etc.

Simphony (Hajjar and AbouRizk 1999) is a computer system for building special purpose construction simulation tools. Different simulation tools were implemented in the Simphony environment (AbouRizk and Hajjar 1998) including AP2-Earth for earthmoving analysis, CRUISER for aggregate production analysis, and CSD for optimization of construction dewatering operations.

2.3 Modeling Techniques of Earthmoving

Different factors are involved in the design of earthmoving operations. These include haulers types, haulers models, haulers number, loaders types, loaders models, loaders number, haul road characteristics (e.g. length, grade, rolling

resistance), etc. Different techniques have been developed to model earthmoving operations, accounting for the factors referred to above. These models can be grouped into: 1) queuing theory models (Halpin and Woodhead 1976, Carmichael 1986-a and 1986-b); 2) linear programming models (Mayer and Stark 1981, Easa 1987); 3) artificial intelligence models, including both expert system models (Christian and Xie 1996, Alkass and Harris 1988, Touran 1990) and neural networks models (Chao and Skibniewski 1994, Flood and Christophilos 1996); 4) regression models (Smith 1999); 5) commercial software models (FPC 1998); and finally 6) simulation models (Oloufa 1993, Shi 1995, McCabe 1997, Martinez 1998, Hajjar and AbouRizk and 1999) .The following subsections review some samples of these models.

2.3.1 Queuing Theory Models

Queuing theory models are considered to be one of the first techniques that have been employed to model earthmoving operations. Halpin and Woodhead (1976) applied finite source queuing models on earthmoving operations by representing trucks as a finite population which enters the system in a cyclic manner. The system boundary embraces the service activity (load) and the queue of trucks (entities) as shown in Figure 2.2. In queuing theory models, the system may be either single-server system (i.e. one loader) or multi-server system (i.e. more than one loader) which service entities (i.e. fleet of trucks). The system changes its states over the time required to complete the task.

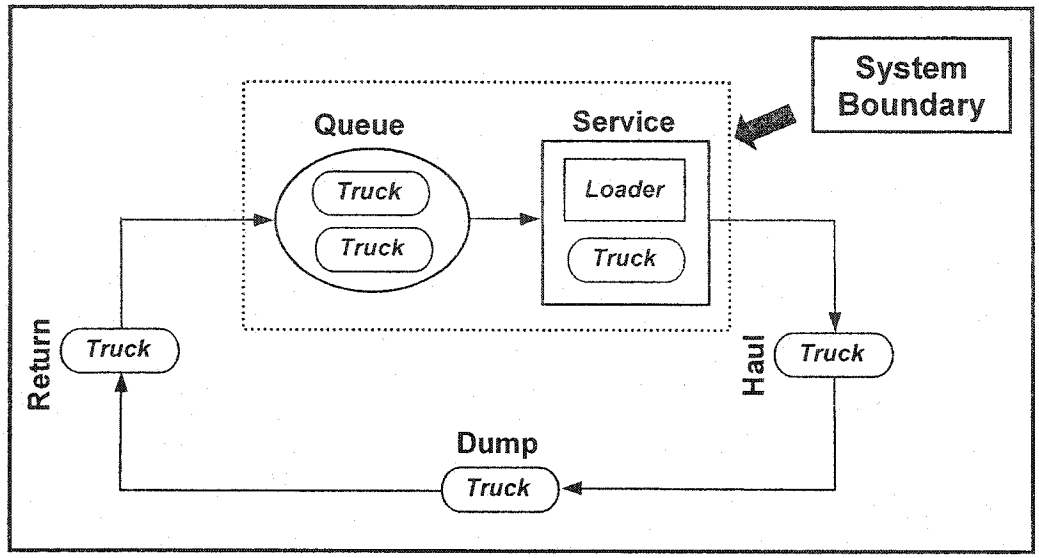


Figure 2.2 System Boundary in Queuing Theory

The number of entities involved in the system defines the number of states that the system may have. For example, an earthmoving operation that has one loader and three trucks, can be modeled with a system that has four states (number of trucks +1). In this case, the arrival of trucks is considered to be proportional to the number of trucks outside the system. Accordingly, the arrival rate of trucks entering the system can be calculated as follows:

Arrival rate = $(M-i)*\lambda$ (2.1)

Where,

M: number of units in the finite population;

i: number of units within the system;

λ: $\frac{1}{T_{av}}$; and T_{av} is average time which a truck stays outside the system.

For the above example, a Makovian model is illustrated in Figure 2.3 where P_i represents the state probability associated with the different states of the system. The term μ is used to represent the average service rate.

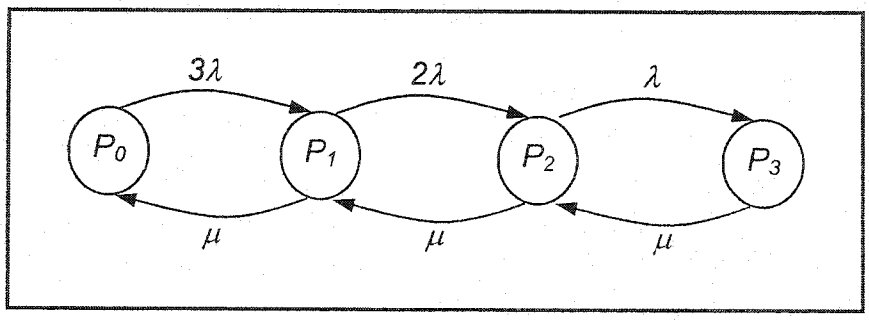


Figure 2.3 Markov Model for Three Trucks and One Loader (adopted form Halpin and Woodhead 1976)

It should be noted that λ and μ represent the transition probability of the system states for ($S_n \Rightarrow S_{n+1}$) and ($S_{n+1} \Rightarrow S_n$) respectively. Equating the inflows and outflows at each node, the equations listed in Table 2.3 are obtained.

The system productivity is calculated making use of an estimated productivity index (P.I., the percent of the time the system contains units) as follows:

$$\text{Productivity} = P.I. * \mu * C \qquad (2.2)$$

Where C is the capacity of the unit being loaded.

Table 2.3 System Nodes States (adopted form Halpin and Woodhead 1976)

Node	Flow Out		Flow In
0	$3\lambda P_o$	=	μP_1
1	$(2\lambda+\mu) P_1$	=	$3\lambda P_o + \mu P_2$
2	$(\lambda+\mu) P_2$	=	$2\lambda P_1 + \mu P_3$
3	μP_3	=	λP_2

Productivity indices can be obtained from different charts (according to whether the system is single or multi-server) using λ, μ and the number of entities. This method assumes that the arrival time of units follows exponential distribution as shown in Figure 2.4.

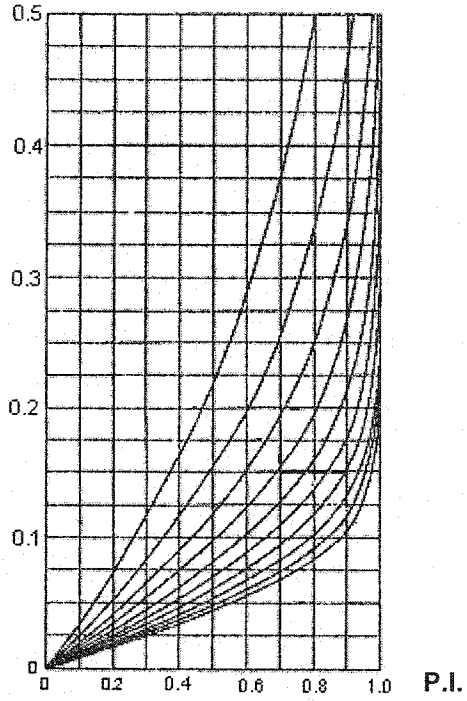


Figure 2.4 Productivity Index Chart (Single-Server System, Halpin and Woodhead 1976)

Also, the mean number of trucks (N) and the average queue length (Q) can be estimated as follows:

$$\overline{N} = \sum_{i=0}^{M} P_i X_i \qquad (2.3)$$

$$\overline{Q} = \sum_{i=0}^{M} P_i (X_i - 1) \qquad (2.4)$$

Where X_i is the number of units in the system associated with P_i.

Carmichael (1986-a) suggested that loading (service) time can be represented by Erlang distribution and that the truck backcycle times follow exponential distribution. These assumptions are based on the observed field data which have been examined with the different queuing model representations (Carmichael 1986-b).

2.3.2 Linear Programming Models

Mayer and Stark (1981) modeled earthmoving operations utilizing linear programming. The unit cost of moving earth is divided into three portions that represent unit cost for excavations, haul and embankment preparation. The unit cost for haul was considered to be proportional to haul distance without any reference to the impact for the type of soils on these operations. Additional cost was added in the case of borrowing earth to represent the cost of purchase. The objective function was set to minimize the total cost of earthmoving as follows:

$$\text{Min } Z = \sum_i \sum_j C(i,j)X(i,j) + \sum_i \sum_j C_D(i,k)X_D(i,k) + \sum_i \sum_j C_B(p,j)X_B(p,j) \qquad (2.5)$$

Where,

$X(i, j)$: the volume of cut earth to be hauled from section i to section j;

$X_D(i, k)$: the volume of cut earth from section i for disposal in section k;

$X_B(p, j)$: the volume borrowed from pit p to be placed in section j;

$C(i, j)$: the unit cost to haul earth from section i to section j;

$C_D(i, k)$: the unit cost to dispose earth from section i to section k;

$C_B(p, j)$: the unit cost to borrow earth from pit p to section j;

It should be noted that different sections are represented by the center of mass for cut and fill zones. The constraints for the objective function are:

$$\sum_j X(i,j) + \sum_k X_D(i,k) = Q_c(i) \tag{2.6}$$

$$\sum_i X(i,j) + \sum_p X_B(p,j) = Q_f(j) \tag{2.7}$$

Where $Q_c(i)$ and $Q_f(j)$ represent the quantity of cut required for section i and the amount of fill required in section j.

$$\sum_i X_D(i,k) \le Q_D(k) \tag{2.8}$$

$$\sum_p X_B(p,j) \le Q_B(p) \tag{2.9}$$

Where $Q_D(k)$ and $Q_B(p)$ represent the capacity of landfill k and the total borrow available in pit p.

$$X(i,j) \ge 0,\ X_D(i,k) \ge 0 \text{ and } X_B(p,j) \ge 0. \tag{2.10}$$

The first two constraints address the limitations of the required quantities of cut and fill. The next two constraints address the capacity of landfills and borrow pits. The last constraint is essentially to ensure the non-negative conditions for all the decision variables and to avoid trivial solutions. Further, the model accounts for the swell and shrinkage of soil by considering their corresponding factors in the previous equations.

Easa (1987) proposed a linear programming method (EARTHN) that considers the variation of unit cost with the quantity of earth moved in a stepwise manner as shown in Figure 2.5. The function may have one or two sudden drops, depending on whether the variation of the total unit cost is due to one portion (purchase or excavation) or two (purchase and excavation). Jayawardane and Harris (1990) developed a linear programming model that incorporates project duration. The model accounts for different practical considerations such as soil strata, swell/shrinkage factors and equipment availability.

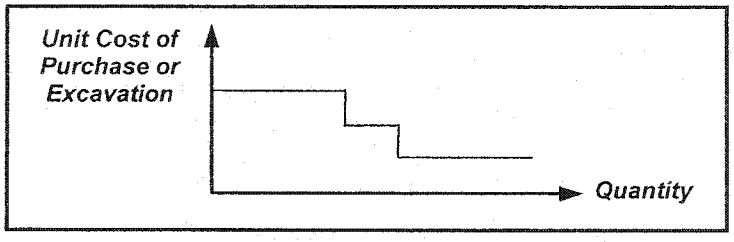


Figure 2.5 Unit Cost versus Quantity of Moved Earth (Easa 1987)

2.3.3 Artificial Intelligence Models

Different earthmoving models have been developed employing artificial intelligence including expert systems and neural networks. Expert systems have been used to select equipment and estimate output rate (Alkass and Harris 1988,

Christian and Xie 1996) or to select equipment and later predict operation output, employing simulation software (Touran 1990). On the other hand, neural networks have been employed either using static-based modeling (Chao and Skibniewski 1994) or dynamic-based modeling (Flood and Christophilos 1996).

Alkass and Harris (1988) developed an expert system for earthmoving plant selection (ESEMPS) for road construction. In order to obtain the details of the selected machines, the user of the system goes through four steps: 1) identifying the task and defining the job conditions; 2) selecting machines by broad category; 3) estimating output and providing machine-matching; 4) selecting the machines to perform the task.

Christian and Xie (1996) developed a prototype knowledge-base expert system to select the most appropriate fleet of equipment and estimate its cost. They conducted a survey among earthmoving contractors in Canada and the United States in order to determine the factors that affect earthmoving operations. They categorized these factors into three groups including: 1) machine selection; 2) production; and 3) cost. The *first* group is subdivided into different factors including the type of operation, company equipment (owned or hired), haul distance, equipment types (single use or multi-use) and type of material. For the *second* group, they considered the restrictions which result from schedule constraints, conflicts with other activities and obstructions. In the calculation of machine output in case of restriction they considered two situations: 1) restrictions are encountered, but the machines are still utilized; and 2) restrictions

are encountered, and the machines are not utilized. For the *third* group, they recommended establishing historical hourly cost data for each individual machine. Based on expert knowledge gathered from questionnaires, they developed a prototype expert system. The user of the prototype goes through three different steps: 1) selects appropriate machines; 2) obtains the production rates and costs; and 3) compares possible alternatives.

Touran (1990) integrated simulation with expert systems (composed of two components) and developed a system to provide the user with the most appropriate type of equipment for an earthmoving operation. The user interacts with the system through different stages. *First*, the user answers different questions that define the job conditions. Subsequently, the first expert system suggests the type of equipment to perform the job. *Second*, the user selects the equipment models and specifies their number, haul and return time to generate the simulation file. *Finally*, another expert system calls INSIGHT simulation package. Subsequently a report is generated which contains all the information pertaining to daily output and cost per unit.

Chao and Skibniewski (1994) proposed neural networks for estimating productivity of earthmoving operations. They applied their approach to earthmoving operations by creating two networks. The first network estimates the excavator capacity according to the job conditions (e.g. depth of cut, type of soil, etc.), whereas, the second estimates the operating efficiency for given operating conditions (e.g. number of trucks, haul distance, grade, etc.).

Flood and Christophilos (1996) modeled earthmoving operations utilizing neural networks. They proposed a methodology to overcome the limitations of static-based neural networks, such as: 1) inability to provide a measure of uncertainty; and 2) ignorance of the non-linear time-wise variance. For the *first* limitation, they suggested the extension of the static-based neural networks into two different ways: 1) including a measure of output variance (e.g. standard deviation of the production rate) as one of the network outputs as shown in Figure 2.6; or 2) including an input parameter which reflects the performance exceedance probability as shown in Figure 2.7.

To overcome the second limitation, they proposed the dynamic modeling technique by creating a recursive loop network which feeds some of its output parameters back as input (see Figure 2.8) and continues until reaching the termination condition(s). Throughout the different runs (recursions), the recursive parameters are initialized by zero values and limited with specified values.

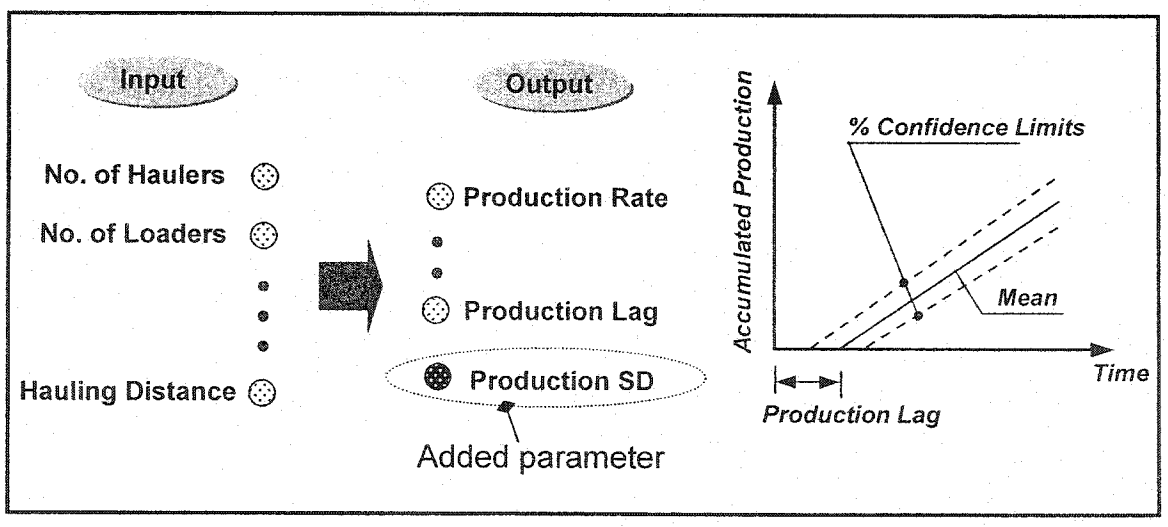


Figure 2.6 Adding Output Parameters (Flood and Christophilos 1996)

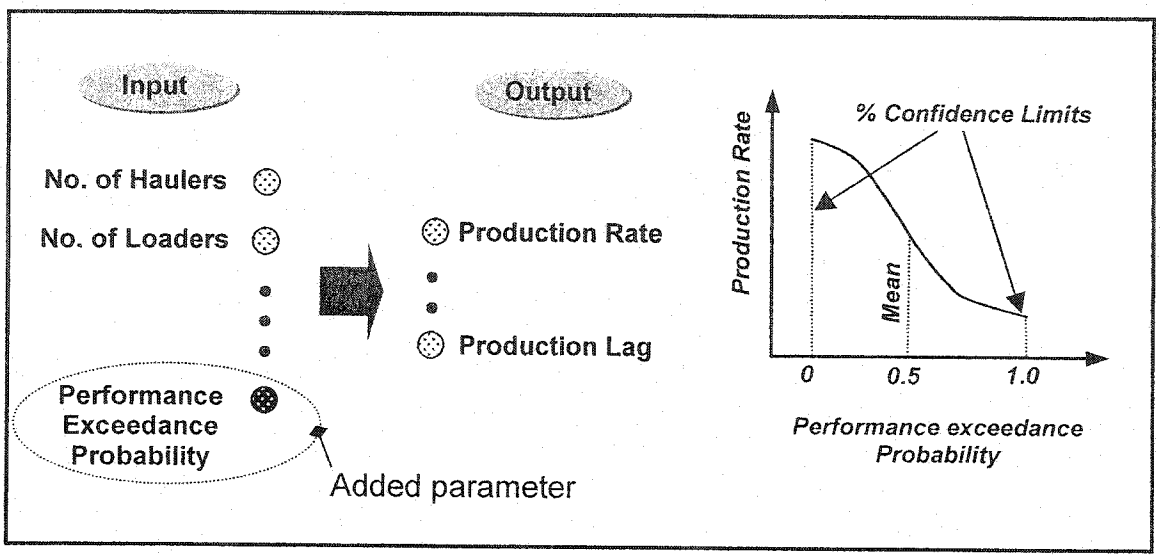


Figure 2.7 Adding Input Parameters (Flood and Christophilos 1996)

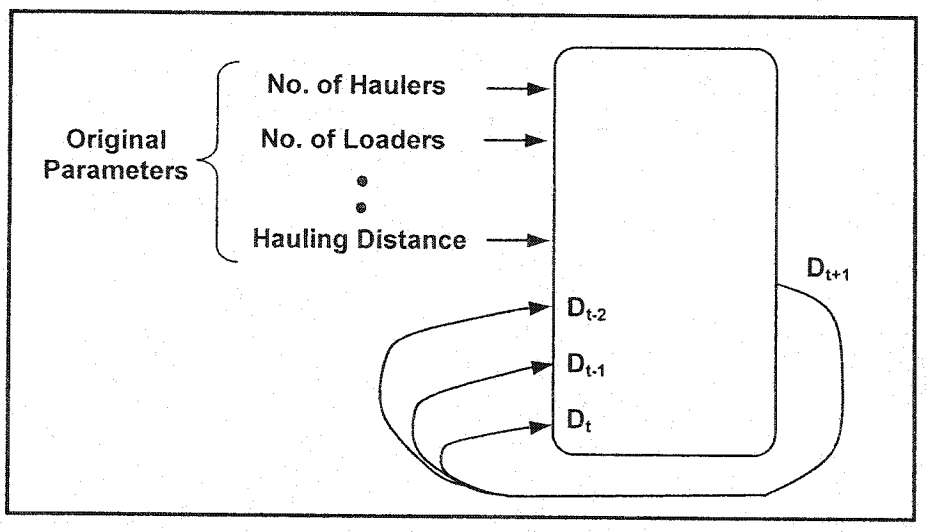


Figure 2.8 Recursiveness of Parameters (Flood and Christophilos 1996)

In their model they considered the dump interval (the interval between the current time in the run and the start time of the next time dump activity) as recursive parameters in earthmoving operations.

2.3.4 Regression Models

Smith (1999) utilized stepwise multiple regression to estimate the productivity of earthmoving operations. Four earthmoving projects were studied with a total of 141 observations. Fourteen characteristics have been observed and calculated. These characteristics are further grouped into three categories as shown in Table 2.4. These factors are: 1) explanatory variables (input of the regression model); 2) interaction variables between two characteristics (input of the regression model); and 3) responses (output of the regression model). A stepwise regression was carried out in order to obtain the factors that have the most significant effect in calculating the actual productivity. The developed regression model does not consider the impact of the material itself (i.e. the type of soil). These factors are listed in Table 2.5 along with their coefficients.

Table 2.4 Factors Influencing Earthmoving Operations (Smith 1999)

Explanatory Variables	Interaction Variables	Responses
Haul length	Haul length/trucks	Actual production
Trucks	Haul length/match factor	Bunch factor
Loaders	Haul length/travel bucket per load	
Match factor	Haul length/travel time	
Load cycle	Trucks/travel time	
Buckets per load	Match factor/travel time	
Travel time	Load cycle/bucket per load	
Bucket volume		
Job		

Table 2.5 Coefficients of the Regression Model (Smith 1999)

Explanatory Variables	Coefficient
Trucks	84.787
Buckets per load	36.806
Bucket volume (m^3)	151.680
Intercept	-297.877
Haul length/trucks	-0.016
Haul length/travel time	9.448×10^{-5}
Match factor	-110.517
Travel time(S)	-0.267
Haul length/match factor	0.107
Haul length(m)	-0.081
Match factor/travel time	-0.244

2.3.5 Commercial Software

Habib (1996) described the cost estimating process of heavy construction, particularly earthmoving operations. The process consists of seven main steps: 1) establish the project's facts; 2) take-off quantities and establish construction methods; 3) select construction equipment; 4) calculate direct cost; 5) calculate indirect cost; 6) establish margins; and finally 7) establish preliminary schedule and cash flow projections. In calculating direct cost, he suggested the use of Fleet Production and Cost analysis software (FPC 1998). This software has been developed by Caterpillar Inc., for estimating productivity, cost and time required for different fleet equipment which moves earth from one place to another over one or more courses (haul road conditions including distance, rolling resistance, grade and restricted speeds). FPC does not consider the interaction between resources (loaders) and entities (haulers). FPC estimates the productivity of a

given operation to be the smaller of the production figures of either the haulers or loaders.

Figure 2.9 illustrates the productivity estimation of an earthmoving operation which consists of one loader and two haulers (as will be described later in the numerical example included in Chapter 3). The operation productivity is calculated as follows:

Operation Productivity = Smallest Prod. * Operator Efficiency * Fleet Availability

= 317.82 * 0.95 * 0.9024 = 272.46 Ton/hr.

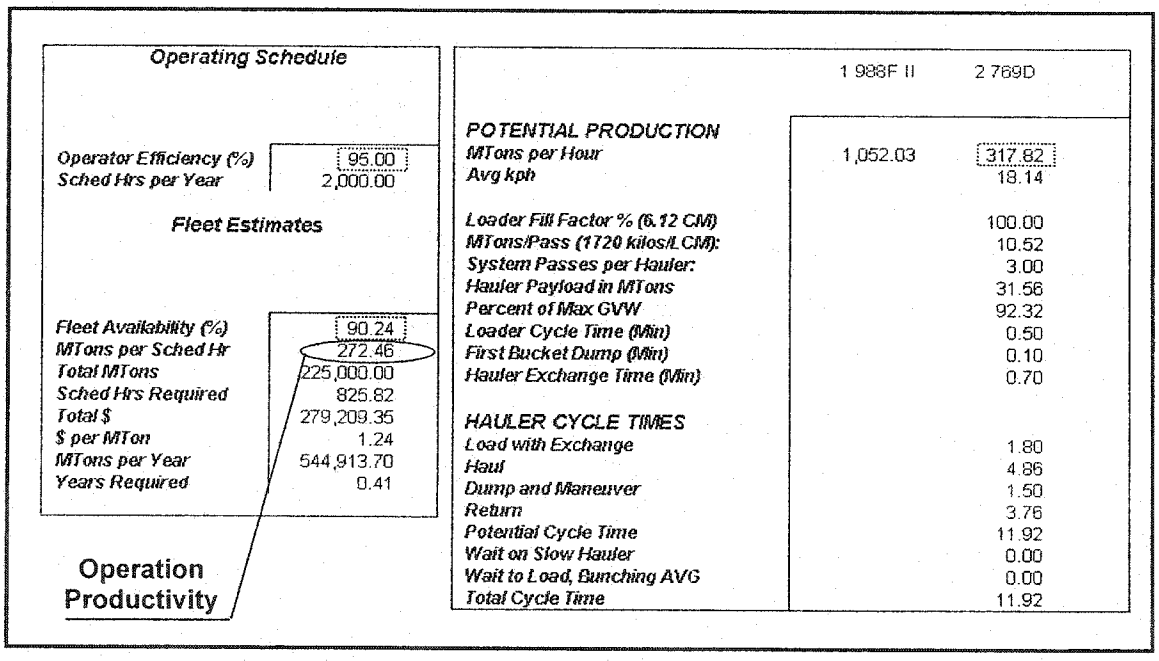


Figure 2.9 FPC Productivity Estimation (FPC 1998)

2.3.6 Simulation Models

Simulation models have been developed to account for uncertainties involved in earthmoving operations. These models include: 1) object-oriented simulation (OOS) models (Oloufa 1993); 2) resource base modeling (Shi 1995); and 3) special purpose simulation (SPS) (Martinez 1998, Hajjar and AbouRizk 1999) Oloufa (1993) implemented a simulation language (MODSIM), designed using object oriented simulation. The user of MODSIM is restricted to a set of specified activities and equipment configurations. For adding activities or changing equipment configurations, the existing code has to be revised in order to account for such changes.

Shi and AbouRizk (1998) proposed an automated modeling system for simulating earthmoving operations (RBM-earth). The user is required to select and link different r-processes involved in the operation at hand. Subsequently, equipment are assigned to the predefined r-processes. RBM-earth uses the SLAM II (Pritsker et al 1997) simulation engine for implementation, and accordingly follows its requirements in modeling and procedures. Shi (1995) listed different optimization algorithms that have been applied to the developed system, implemented using MicroCYCLONE software (different from the one used for the system). These optimization algorithms are performed considering only the change in the quantities of the involved entities and associated resources. They do not account for the utilization of alternative models for those entities and resources.

McCabe (1997) integrated both simulation (SLAM II) and belief networks (as a diagnostic tool) to obtain a near optimum fleet scenario, accounting for both the quantity and the capacity of servers (resources) and customers (entities). Different remedial actions can be obtained based on the specified limits of a set of predefined indices (QL, QW, SQ, SU and CD). The limitations of this model are that: 1) remedial actions may configure a fleet that is not available, since earthmoving contractors are usually limited with a specific set of equipment, whether owned or rented; 2) a lot of efforts are required to estimate the limits of the different indices; and 3) It ignores project indirect cost.

Martinez (1998) developed a special purpose simulation tool (EarthMover), designed for earthmoving operations. The user is required to define different inputs to EarthMover including loading and hauling equipment and characteristics of different road segments by dragging the corresponding elements from the available graphical interface. Different software is used in EarthMover: *Visio* (1997) for graphical input entry; *MS Visual Basic* for non-graphical input entry; *STROPOSCOPE* (Martinez 1996) as a simulation engine; *MS Excel* to interpret simulation output; and *Proof* for animation. EarthMover lacks the ability to provide hourly owing and operating costs for the equipment used, needed for calculating the total cost of an operation. It also lacks optimization capabilities to advise on the selection of a near optimum fleet configuration.

Hajjar and AbouRizk (1999) proposed the Simphony as an environment for building special purpose simulation tools. Different simulation tools have been

built in the Simphony which are utilized as templates for different applications. One of these templates has been designed for earthmoving operations. This template is used as a special purpose simulation tool that has an interactive user interface that eases modeling of simulation processes. Three limitations have been found in Simphony:

1) Haulers' travel time is calculated with an approximate procedure without considering the acceleration and de-acceleration in transition zones between the road segments (different total resistances). This leads to inaccurate estimation of travel time. For example, the following equation (Simphony 2000) is used to calculate the haulers' speed:

$$Speed = TopSeed(Loaded/Empty) * \frac{(55 - TotalResistance)}{55} \qquad (2.11)$$

Accordingly, the travel time across the section is calculated based on the obtained speed and the segment length;

2) It lacks the ability to analyze different scenarios at the same time and to advise on selection of a near optimum fleet configuration; and

3) It requires modifications by developers to modify the existing code in order to account for new variations (e.g. adding compacting activity).

2.4 Simulation Optimization

Optimizing total cost of earthmoving projects represents a challenging task. There are many factors related to quantity of resources, type of equipment used and indirect cost involved in the optimization process of these projects. To optimize a project is to find the solution that provides the best use of resources in an effort to either minimize the project total duration or its total cost.

Simulation optimization is defined as the process of maximizing information retrieval from simulation analysis without carrying out the analysis for all combinations of input variables (Carson and Maria 1997). A number of simulation optimization techniques have been described in literature (Azadivar 1992, Carson and Maria 1997, Azadivar 1999). These techniques have been applied in different manufacturing processes (Azadivar et al 1996, Hall et al 1996). Figure 2.10 shows the simulation optimization methods as proposed by Carson and Maria (1997).

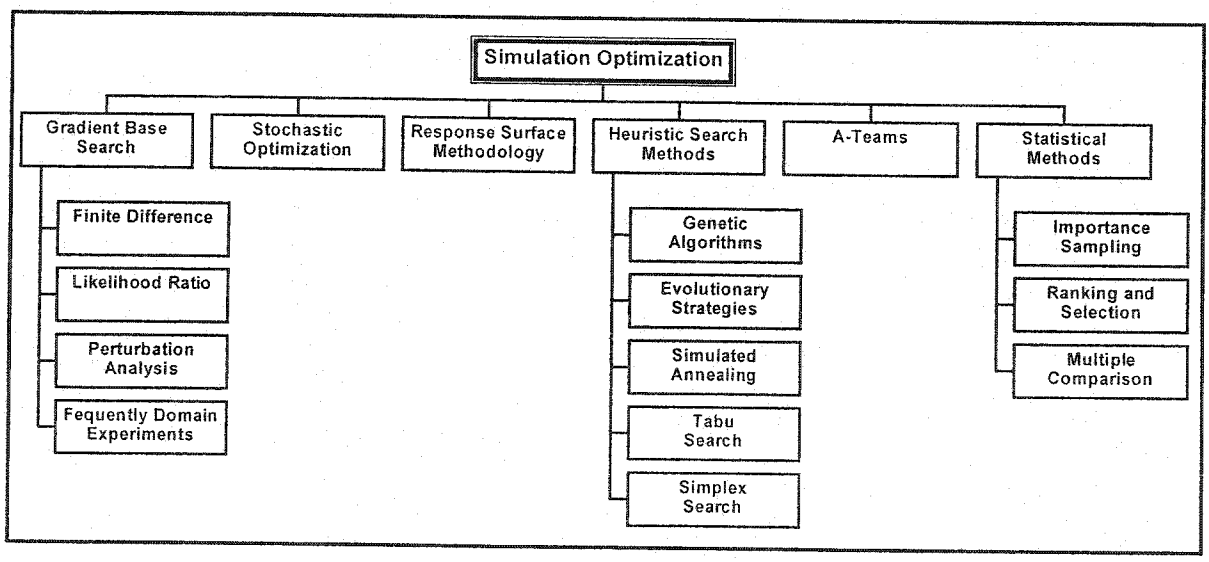


Figure 2.10 Simulation Optimization Methods (Carson and Maria 1997)

An efficient optimization model is the one that accounts for both qualitative (e.g. type of resources, their combinations, etc.) and quantitative (e.g. the number of utilized resources) variables. Figure 2.11 illustrates the interaction process between simulation and optimization models. In general, the process starts by defining the variables (both qualitative and quantitative) to the optimization model which in turn configures a set of inputs parameters that depict the characteristics of the system being simulated. This set is used as input for the simulation model. Subsequently, simulation runs are performed and their output is passed to the optimization model to evaluate the degree of satisfying the objective function. If the configured system does not satisfy the requirements, a new system is configured and the loop is repeated, otherwise, the process is terminated.

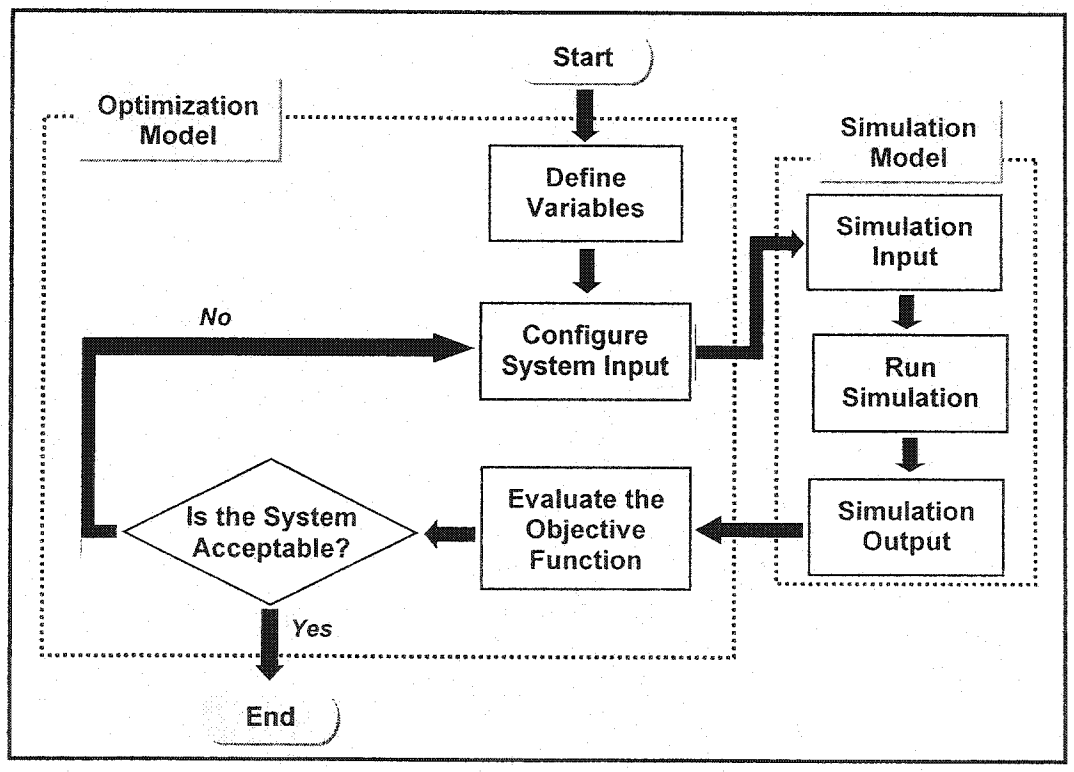


Figure 2.11 Interaction between Optimization and Simulation Models

Simulation optimization did not receive similar attention in construction. Some efforts, however, have been made in that respect (AbouRizk and Shi 1994, Shi and AbouRizk 1995, McCabe 1998). These efforts do not account for the indirect cost portion as a decision variable that influences selection of equipment fleets.

Genetic algorithms (GAs) (Goldberg 1989, Holland 1992, Coly 1999) have been used as a powerful tool for optimization. They are essentially heuristic search techniques which follow random sampling. GAs have been coupled with simulation to achieve different optimizations objectives (Baesler and Sepulverda 2000, Boesel et al 1999, Azadivar and Tompkins 1999). In construction industry, GAs were utilized for: 1) estimation of optimum markup (Moselhi et al 1993-a), 2) resource scheduling (Hegazy 1999-a and b, Li and Love 1997, Feng and Liu 1997, Chan et al 1996), 3) site layout (Philip et al 1997), and 4) maintenance budget allocation and pavement rehabilitation decisions (Chan et al 1994, Fwa et al 1995). GAs were also used by Haidar et al (1999) to optimize equipment selection in opencast mining. The model, however, neither accounts for qualitative variables nor indirect cost.

2.5 Summary

This chapter reviewed simulation modeling techniques including discrete, continuous and combined simulation have been reviewed. It also outlined the different aspects of applying simulation in construction: 1) development of simulation languages such as CYCLONE (Halpin 1977) and STROPOSCOPE

(Martinez 1996); 2) application of object-oriented simulation techniques including MODSIM (Oloufa 1993) and CIPROS (Tommelein et al 1994); 3) application of hierarchical simulation modeling (Sawhney and AbouRizk 1995); 4) application of resource-based modeling (Shi 1995; Oloufa et al 1998); 5) integration with belief networks (McCabe 1996); and 6) application of special purpose simulation (AbouRizk and Hajjar 1998, Martinez 1998).

Modeling techniques for earthmoving operations have been reviewed, including 1) queuing theory models (Halpin and Woodhead 1976); 2) linear programming (Mayer and Stark 1981, Easa 1987); 3) artificial intelligence models including both expert system models (Christian and Xie 1996) and neural networks (Flood and Christophilos 1996); 4) regression models (Smith 1999); 5) commercial software models (FPC 1998); and finally 6) simulation models (Oloufa 1993, Shi 1995, McCabe 1997, Martinez 1998, Hajjar and AbouRizk 1999). Reviewed also is the interaction between simulation and optimization models along with methods used for simulation optimization.

CHAPTER 3

PROPOSED SIMULATION MODEL

3.1 General

This chapter presents the layout of the proposed system and describes briefly its basic components. It focuses primarily on the development of the earthmoving simulation program (*EMSP*) which represents the simulation engine of the developed system (*SimEarth*). It also presents the modeling aspects of the simulation process utilizing discrete event simulation, three-phase simulation and object-orientation features. These features include: 1) classes; 2) dynamic data structure; 3) inheritance and 4) polymorphism.

Compared to other simulation models, *EMSP* has a number of interesting features (Marzouk and Moselhi 2000): 1) it does not require either programming from the end-users or familiarity with simulation languages; 2) it provides an optimization tool geared towards the selection of a near optimum fleet, accounting for equipment available to contractors; 3) it provides a more realistic estimate of haulers' travel time; and 4) it makes full use of object-orientation features.

A numerical example of an actual case is analyzed to validate the developed simulation engine and demonstrate its capabilities. The results are compared to those generated using Caterpillar software (*FPC*). The engine and *FPC*

recommend the same fleet and their estimated project durations are very close, with a difference of less than 8%. Unlike *FPC*, the developed engine, however, can model and account for uncertainty, in a reliable manner, during the execution of earthmoving operations.

3.2 System Layout

A newly-developed simulation system (*SimEarth)* has been designed and implemented in MS environment as a special purpose tool for estimating time and cost of earthmoving operations. *SimEarth* user interface has been coded using Visual Basic 6.0 and designed to integrate its various components (see Figure 3.1). It consists of: 1) EarthMoving Simulation Program (*EMSP*); 2) Equipment Cost Application (*ECA*); 3) Equipment Database Application (*EDA*); 4) EarthMoving Genetic Algorithm (*EM_GA*) and 5) Output Reporting Module (*ORM*). Beside these main components, *SimEarth* is supported by: a) Hauler's Travel Time Application (*HTTA*) and b) EarthMoving Markup Application (*EMMA*). The *ECA, EDA, HTTA* and *EMMA* components are designed for use as stand-alone applications or as integrated applications within *SimEarth*. The characteristics of those components, their functions and their respective implementation environment are briefly summarized in Table 3.1. All system components are implemented in MS environment except the dynamic sub-module of *ORM* component, which is implemented utilizing Proof Animation (2000) Software. Each individual component has been tested and validated to ensure that it performs as intended. The overall performance of *SimEarth* is

validated via an actual case study (see Chapter 7) to ensure the overall system performance and data flow among its basic components.

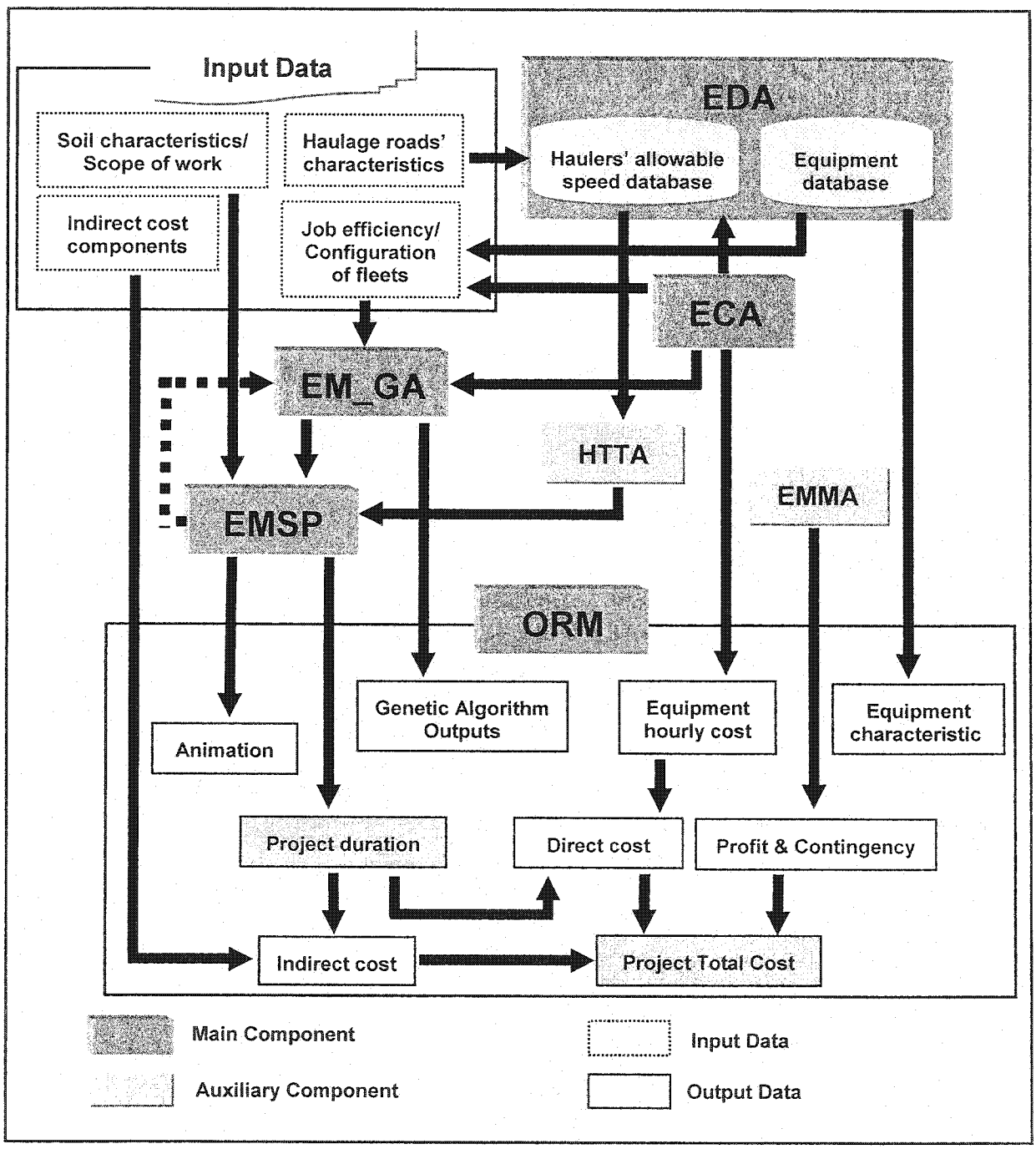


Figure 3.1 Components of SimEarth

Table 3.1 Characteristics of the Systems' Components

Component	Characteristics	Implemented Environment
EMSP	It represents the main component of the developed system. It performs simulation replication(s) to a predefined set of resources and entities involved in different experiment(s). It interacts dynamically with all system's components except *EMMA*.	MS Visual C++ 6.0
ECA	It is a spreadsheet application, dedicated to estimating the hourly owning and operating costs of any earthmoving equipment. It interacts with *EMSP*, *EDA* and *ORM*.	MS Excel 97
EDA	It is a relational database application, designed to provide equipment related information including their characteristics, attachments, range dimensions, etc. It interacts with *EMSP*, *ECA*, *HTTA* and *ORM*.	MS Access 97
EM_GA	It is an optimization algorithm, designed to select near-optimum fleet configuration that yields least total project cost. It interacts with *EMSP* and *ORM*.	MS Visual C++ 6.0
ORM	It consists of two main sub-modules: static and dynamic. The static sub-module is a spreadsheet that provides all information in paper format related to simulation results, equipment cost breakdown, equipment characteristics, and project estimated total cost and schedule. The dynamic sub-module supports animation to represent the overall system dynamics, the interaction between the customer and server resources, the bottlenecks of the system, and the delay associated with the resources. It interacts with all system's components except *HTTA*.	MS Excel 97 Proof Animation
HTTA	It has been developed to provide realistic estimates of haulers' travel time utilizing fuzzy clustering. It accounts for: 1) the performance of haulers over their life time and 2) acceleration and deceleration in transition zones. It interacts with *EMSP* and *EDA*.	MS Excel 97
EMMA	It is dedicated for estimating markup percentage, which includes profit and contingency margins. It estimates markups, accounting for the risk attitudes of decision-makers. It has been developed utilizing multi-attribute utility theory and analytic hierarchy process. It interacts only with *ORM*.	MS Visual Basic 6.0

3.3 EMSP Design

EMSP has been designed utilizing discrete event simulation (DEVS) (Banks 1998) and object-oriented modeling (Quatrani 1998, Deitel and Deitel 1998,

Skansholm 1997). Different features of object-orientation have been employed including classes, objects, dynamic data structure, and polymorphism. The three-phase simulation (Pidd 1995) was employed instead of process interaction in order to control the dynamics of *EMSP* by tracking the activities of the simulated process. The three-phase simulation approach is considered most appropriate for object oriented simulation (OOS), specially when there are too many entities involved in the process being modeled to avoid synchronicity loss (Pidd 1995). The utilization of three-phase simulation and object-orientation in the developed engine are described subsequently.

3.3.1 The Use of the Three-Phase Simulation

Simulation modeling technique is an initiation of the operation of a real-world process or system over time (Banks 1998). Discrete-event simulation (DEVS) changes occur at discrete time points. Accordingly, there is a need for a mechanism to track these changes with respect to time. Pidd (1984) proposed that the hierarchy of any computer simulation application can be organized using three levels: 1) executive, responsible for advancing simulation time and assuring that activities are performed during the advancement of simulation time; 2) application blocks, responsible for establishing the dynamic logic and interaction among the entities of the application (e.g. process interaction and three phase simulation); and 3) utilities and procedures, used to store and track entities and gather information. Figure 3.2 illustrates the proposed DEVS hierarchy.

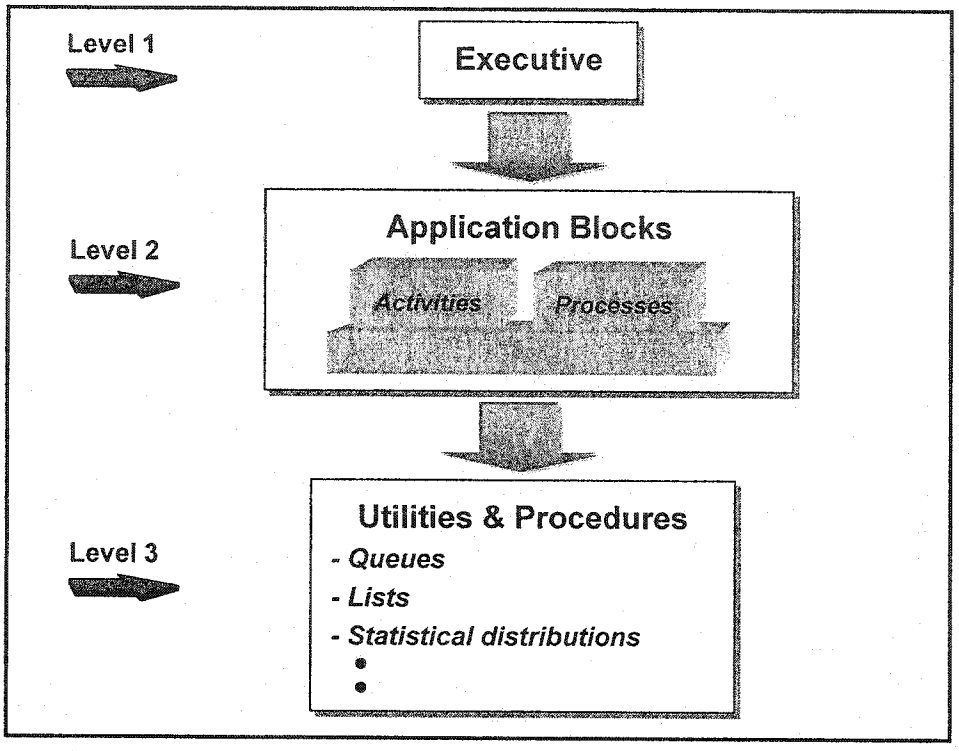


Figure 3.2 DEVS's Application Hierarchy

Most commercially available simulation software systems track entities rather than activities (Banks 1998). This is achieved by defining different lists that store the entities according to their states throughout the simulation process. For example, lists containing current events, future events and delays are used to store entities that have states which are ready, time-delayed and condition-delayed, respectively. On the other hand, using three-phase simulation, *EMSP* tracks an activity such as hauling rather than an entity such as hauling trucks or their payload. In this case the start and finish events of the individual activities provide checkpoints, for guiding the simulation engine in completing its task. Activities are classified into two types (Pidd 1984): 1) bound to happen (Bs) and

2) conditional (Cs) activities. In this case, only one list is used to track both Bs and Cs activities instead of using more than one list to represent the states of the entities involved.

EMSP tracks activities in three phases: phase 1 which advances simulation time to the next simulation event; phase 2 which carries out all due Bs; and phase 3 which carries out all possible Cs. Figure 3.3 depicts a number of activities being tracked by *EMPS* for an earthmoving operation that contains the main activities of load, haul, dump, and return. In Phase 1, *EMSP*'s first activity is removed and the simulation time is advanced to the next time. In Phase 2, all due B activities (bound to happen) are carried out. In Phase 3, all possible C activities (conditional) are performed. It should be noted that an activity (e.g. haul) may exist more than one time since it may be carried out by different entities (e.g. hauler ID = 2 and 3) as shown in Figure 3.3.

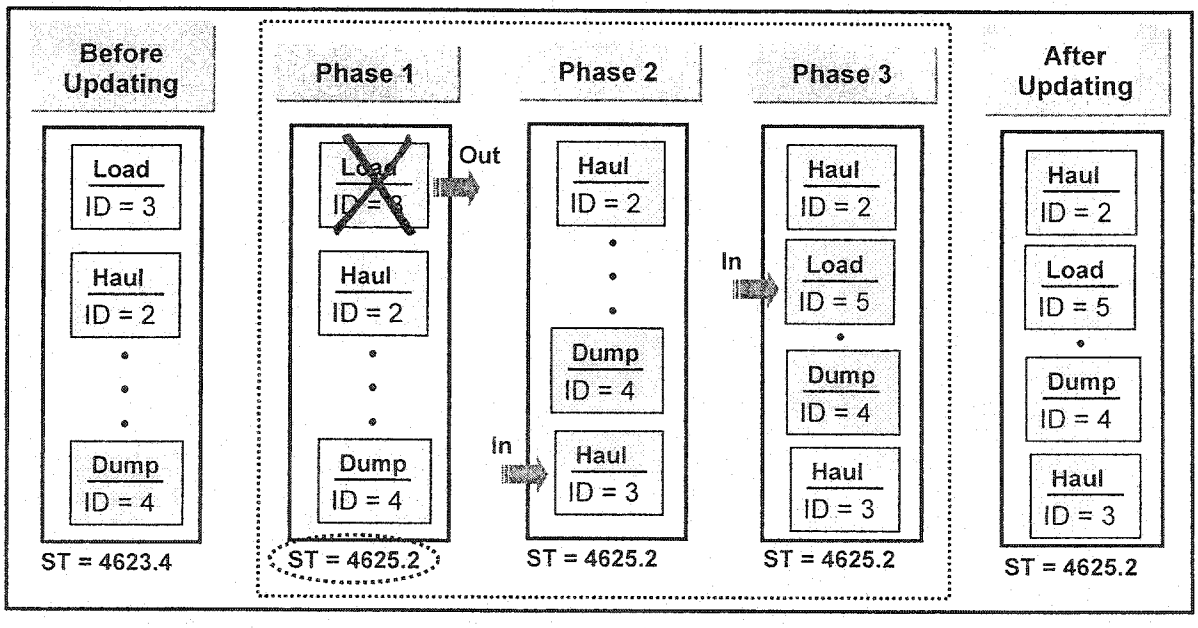


Figure 3.3 EMSP: Tracking Activities

3.3.2 Implementation Using Object-Orientation

In mapping real-world system processes using simulation, two main conditions should be respected: 1) to capture accurate representation; and 2) to simplify the modeling process for the user (Oloufa 1993). OOS models the interaction and the communication among the objects over the simulation time period (Joines and Roberts 1998). The interaction occurs by sending a message from one object to invoke another object to perform a specific task. The message is transferred via data methods in the classes. *EMSP* has been designed using OOS utilizing different features including classes, dynamic data structure and polymorphism (Quatrani 1998, Deitel and Deitel 1998, Skansholm 1997). The use of these features in the development of *EMSP* is described subsequently.

In the design of *EMSP*, different types of classes have been defined to capture the properties of key objects in earthmoving operations (Marzouk and Moselhi 2000). This includes objects of entities, resources, activities and stored simulation statistics. The classes are coded in MS Visual C++ 6.0. Figure 3.4 provides an example of a class named `Piled_Earth`, designed to track the piled earth during the simulation process. The class has three attributes (data members): `Earth_To_Pile;` `Ready_To_Haul` and `Piled_Soil`. The `Earth_To_Pile` attribute reflects the scope of earth to be pilled, whereas `Ready_To_Haul` and `Piled_Soil` attributes represent the amount of piled earth that is ready to be hauled and the amount of piled earth at any simulation time, respectively.

The class also has five methods (member functions) in addition to its constructor (`Piled_Earth()`) which initializes its data members. These methods are: `Set_Earth_To_Pile(..);Increase_PiledEarth(..);Decrease_Piled Earth(..);` `IsHaulPossible(..)` and `IsPileCompleted()`. The `Set_Earth_To_Pile(..)` method is used to set the scope of earth that is required to be piled at the beginning of the simulation process. The `Increase_PiledEarth(..)` method is used to increase the quantity of piled earth that is ready to be hauled (represented by `Ready_To_Haul` attribute) and the amount of piled earth (represented by `Piled_Soil` attribute).

```
class Piled_Earth
 {
 public:
  Piled_Earth(){Piled_Soil=0, Earth_To_Pile=0, Ready_To_Haul=0;} //constructor
  void Set_Earth_To_Pile (double SEarth_To_Pile){Earth_To_Pile=SEarth_To_Pile;}
  void Increase_PiledEarth(double PileEquipCapacity){
        Ready_To_Haul += PileEquipCapacity;
        Piled_Soil += PileEquipCapacity;
  }
  void Decrease_PiledEarth(double HaulerCapacityacity ){Ready_To_Haul -=HaulerCapacity;}
  bool IsHaulPossible (double HaulerCapacity){
    if (Ready_To_Haul < HaulerCapacity)
        return false;
    else
        return true;
  }
  bool IsPileCompleted (){
    if (Piled_Soil < Earth_To_Pile)
        return false;
    else
        return true;
  }
 private:
  double Earth_To_Pile;
  double Ready_To_Haul;
  double Piled_Soil;
 };
```

Figure 3.4 Piled_Earth Class

This increase is equivalent to the pile equipment capacity which is passed as a parameter to the method. Contrary to the `Increase_PiledEarth(..)` method, the `Decrease_PiledEarth(...)` method is used to decrease the amount of piled earth that is ready to haul by passing the hauler capacity as a parameter to the method. The `IsHaulPossible(..)` method is a Boolean function that returns *false* when the amount of earth that is ready to haul is less than the haulers' capacity, otherwise it returns *true*. The `IsPileCompleted()` method is also a Boolean function that returns *true* when all earth has been piled and accordingly, a message is passed to stop the piling activity.

One of the powerful features of object-orientation is dynamic data structure (Deitel and Deitel 1998, Skansholm 1997). This feature enables *EMSP* to store, track and gather information from the objects without knowing their number in advance. The stored objects increase and decrease during a simulation run. Queues and lists have been utilized in *EMSP* as an application of the dynamic data structure. *Queues* are used to insert objects at the end of a list and remove them commencing from the beginning (first in, first out), creating a queue of objects. On the other hand, *lists* are used to insert objects of activities at any location on a list (beginning, middle or end) based on their finish time. The resulting list of activities, called current activity list (*CAL*), is used to track all the activities. In *CAL*, all the activities are organized in a chronological order according to their finish time. For example, in the simulation process depicted in Figure 3.5, there are six trucks involved: one is being loaded (no. 5), two are

waiting in the queue (no. 4 and 6), one is returning (no. 1) and two are hauling (no. 2 and 3) at a specific time in a simulation process. Table 3.2 lists *CAL* activities at that simulation time. It should be noted that the finish time of the first activity in *CAL* is used to reset the simulation clock. For the example described, the simulation clock is set to 2661.3 time units. Later, in order to advance the simulation clock, the first activity in *CAL* will be removed while *EMSP* checks for the possibility of scheduling new B (bound to happen) and C (conditional) activities.

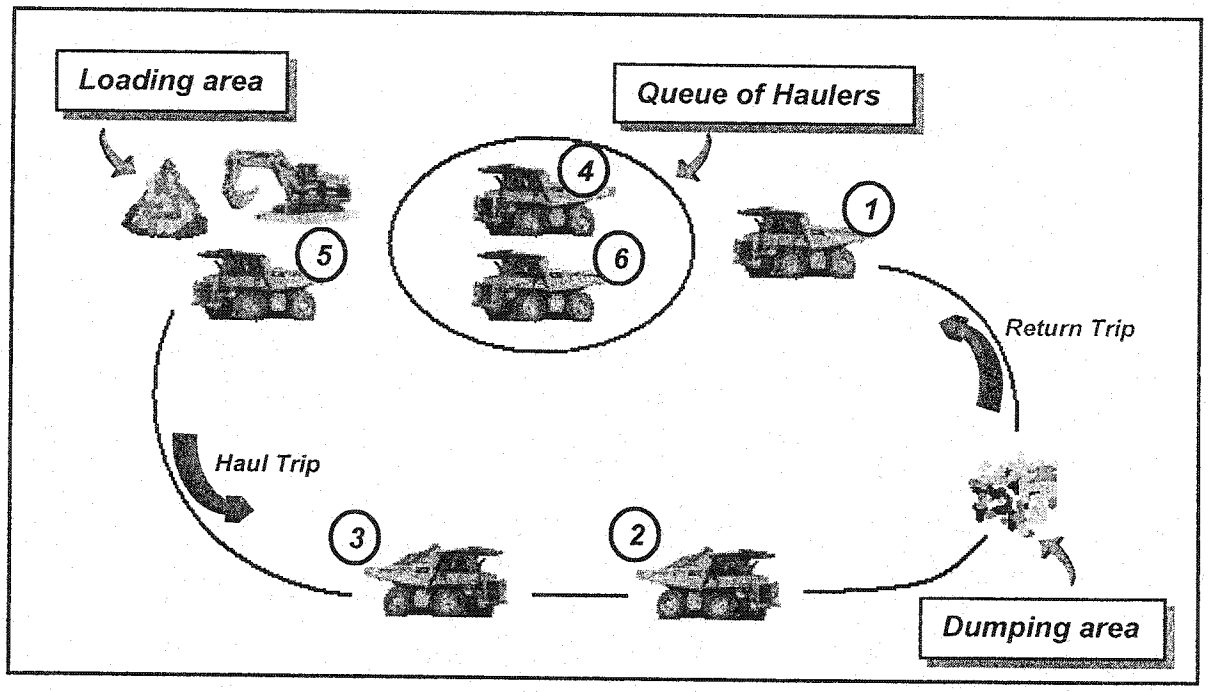


Figure 3.5 Distribution of Trucks before Resetting the Simulation Clock

Table 3.2 CAL Activities before Resetting the Simulation Clock

Hauler	Activity	Start Time	Finish Time
5	*Load*	*2659.7*	*2661.3*
2	*Haul*	*2640.1*	*2665.2*
1	*Return*	*2644.5*	*2670.8*
3	*Haul*	*2650.8*	*2675.6*

To illustrate the method employed in the dynamic of scheduling activities, consider that the check performed by *EMSP* indicates that two activities are about to start: one is B (haul) and the other is C (load). These activities are inserted in *CAL*, according to their chronological order as shown in Table 3.3. Figure 3.6 depicts the redistribution of trucks after resetting the simulation clock to 2662.8 time units.

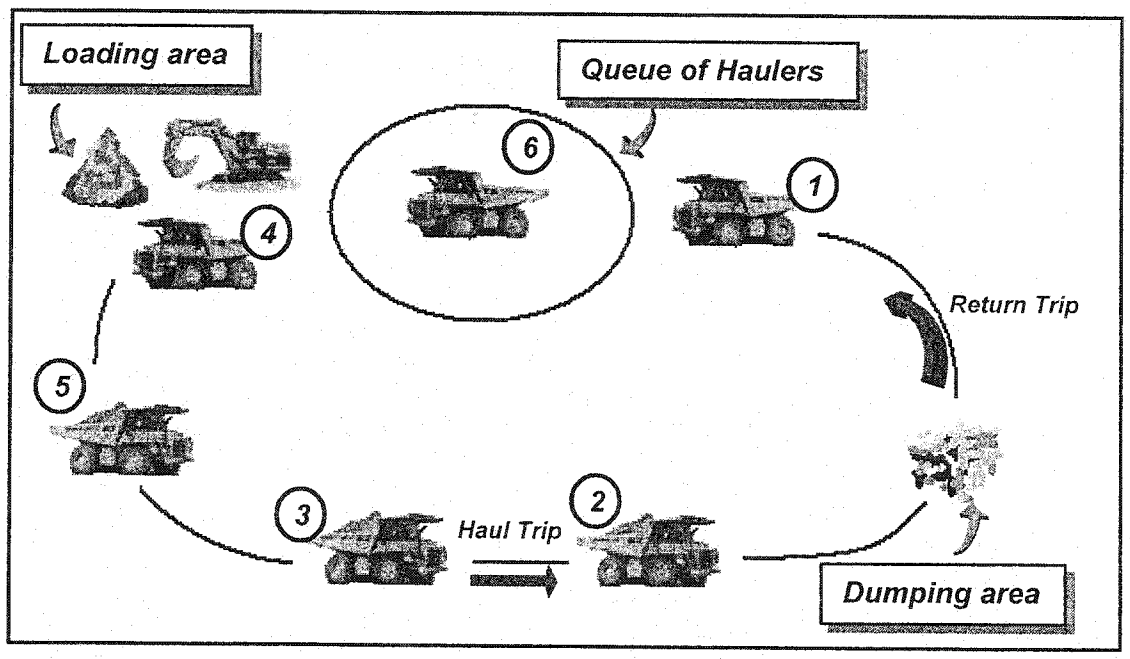


Figure 3.6 Distribution of Trucks after Resetting the Simulation Clock

Table 3.3 CAL Activities after Resetting the Simulation Clock

Hauler	Activity	Start Time	Finish Time
4	Load	2661.3	2662.8
2	Haul	2640.1	2665.2
1	Return	2644.5	2670.8
3	Haul	2650.8	2675.6
5	Haul	2661.3	2686.4

New Schedule

Polymorphism is an object-orientation feature that enables code sharing and reusability (Pidd 1995, Deitel and Deitel 1998,). It has different forms including templates, overloading and overriding. A *template* is a tool for constructing a generic class or function of different types (Skansholm 1997). For example, *EMSP* uses templates in the implementation of class `Queue` by passing the variable `ELETYPE` as a parameter which is utilized in the class although its type is unknown (see Figure 3.7). Accordingly, different queues can be created to store different types such as integers, doubles and objects (e.g. loaders, etc.).

```
template <class ELETYPE>
class Queue
{
public:
        Queue();     // constructor
        ~Queue();      // destructor
        .
};
```

Figure 3.7 Queue Template Class

Overloading occurs when there are different functions with different codes having the same name. The desired function is invoked based on the attributes which are passed to it. Overloading is employed by *EMSP* in different situations. For example, the `M_Simulate` class has two methods with the same name `Output_Statistics(int, double**, fstream&fout)` and `Output_Statistics (fstream&fout)`. The first one performs statistical analysis for the simulation outputs which are stored in two-dimensional arrays and then prints the results to an output file, whereas, the second one prints null values to a specified output file.

Overriding is the third form of polymorphism which occurs when one or more methods in a sub-class share the same name which has been implemented with its super-class(es). The implemented method in the sub-class hides the method(s) having the same name in the super-class(es). It is utilized by *EMSP* in the `OPY_Simulate` class and its sub-classes which have a method called `Activity_Drive()`. A created object for sub-classes of the `OPY_Simulate` class utilizes its own `Activity_Drive()` method, ignoring the one(s) implemented in its super-class(es).

3.3.3 EMSP Dynamic Data Flow

EMSP is coded using MS visual C++ 6.0 and runs under the control of the developed *SimEarth* system. *SimEarth* is implemented using MS visual basic 6.0 to facilitate the design and development of the user interface screens. *EMSP* receives its input from *SimEarth* in two forms (Figure 3.8): 1) passing arguments to its main function; and 2) reading from external files. Eight parameters are passed to *EMSP*'s main function: 1) simulation performed either for a test mode or an analysis mode, 2) condition(s) of termination for the simulation analysis, 3) interaction among equipment in the fleet under consideration, 4) storage of simulation data for later use in animation, 5) selected fleet scenario for simulation analysis, 6) existence of second hauler or not, 7) the selected set of activities involved in the simulation process and 8) number of simulation runs.

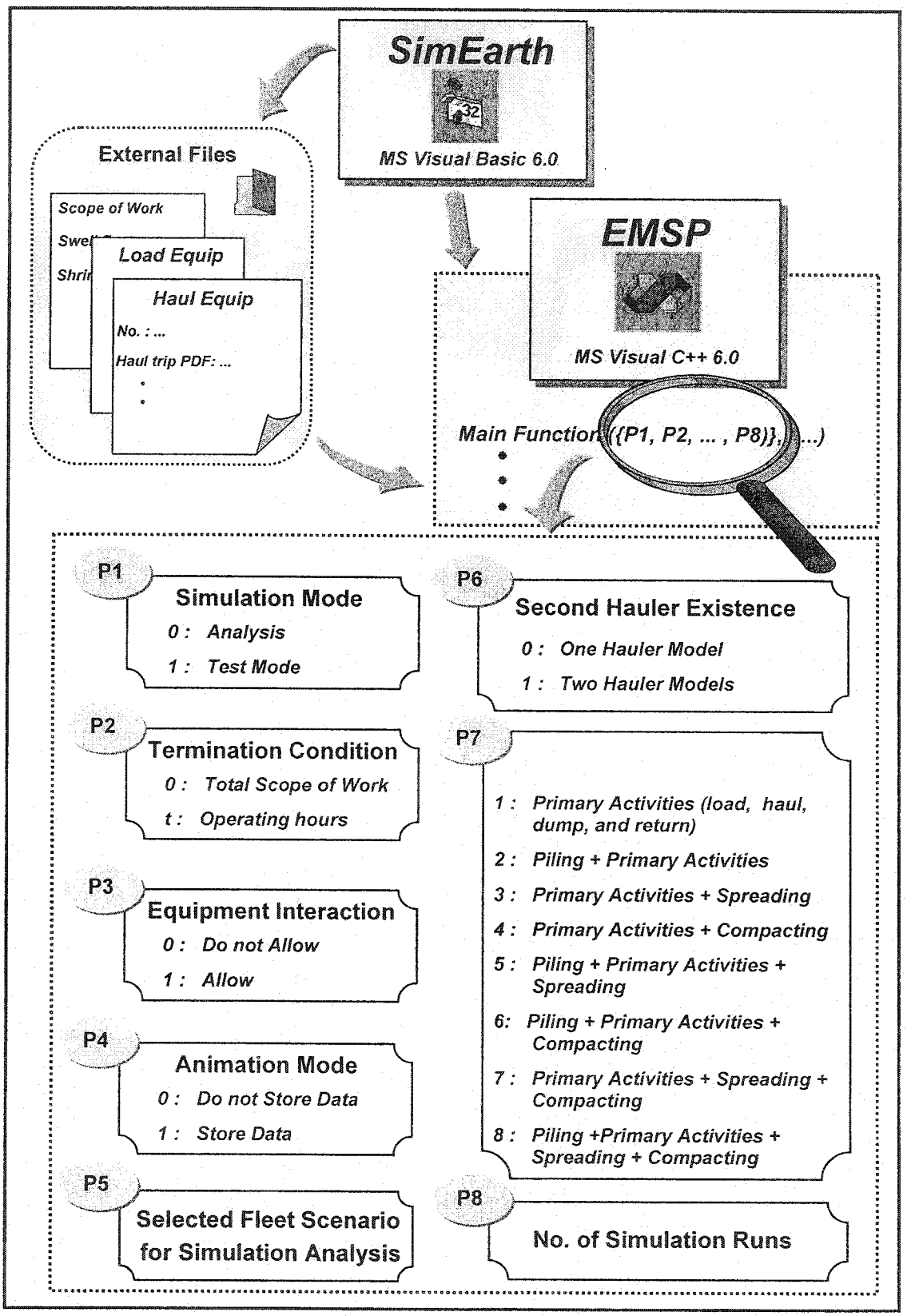


Figure 3.8 Schematic Diagram for EMSP Input Data

All data pertinent to scope of work, equipment characteristics such as equipment, equipment productivity and its related probability density functions are passed to *EMSP* through external input files. These files are saved in text format files. Figure 3.8 provides a schematic diagram, illustrating how data are transferred from *SimEarth* to *EMSP*.

The classes used in the design of *EMSP* are of two types: auxiliary and main (see Figure 3.9). Auxiliary classes are connected to the main classes through either association or aggregation relationships, whereas, the main classes are connected to each other through inheritance relationships. The main classes of *EMSP* capture different situations according to the activities involved. Therefore, they represent different combinations of earthmoving activities. Table 3.4 lists all main classes along with their corresponding activities.

Table 3.4 EMSP Main Classes and Corresponding Activities

Class	Corresponding Activities
OPY_Simulate	Load, dump, haul, and return
OPE_Simulate	Piling, load, dump, haul, and return
OSD_Simulate	Load, dump, haul, return, and spreading
OCT_Simulate	Load, dump, haul, return, and compacting
PS_Simulate	Piling, load, dump, haul, return, and spreading
PC_Simulate	Piling, load, dump, haul, return, and compacting
SC_Simulate	Load, dump, haul, return, spreading, and compacting
PSC_Simulate	Piling, load, dump, haul, return, spreading, and compacting

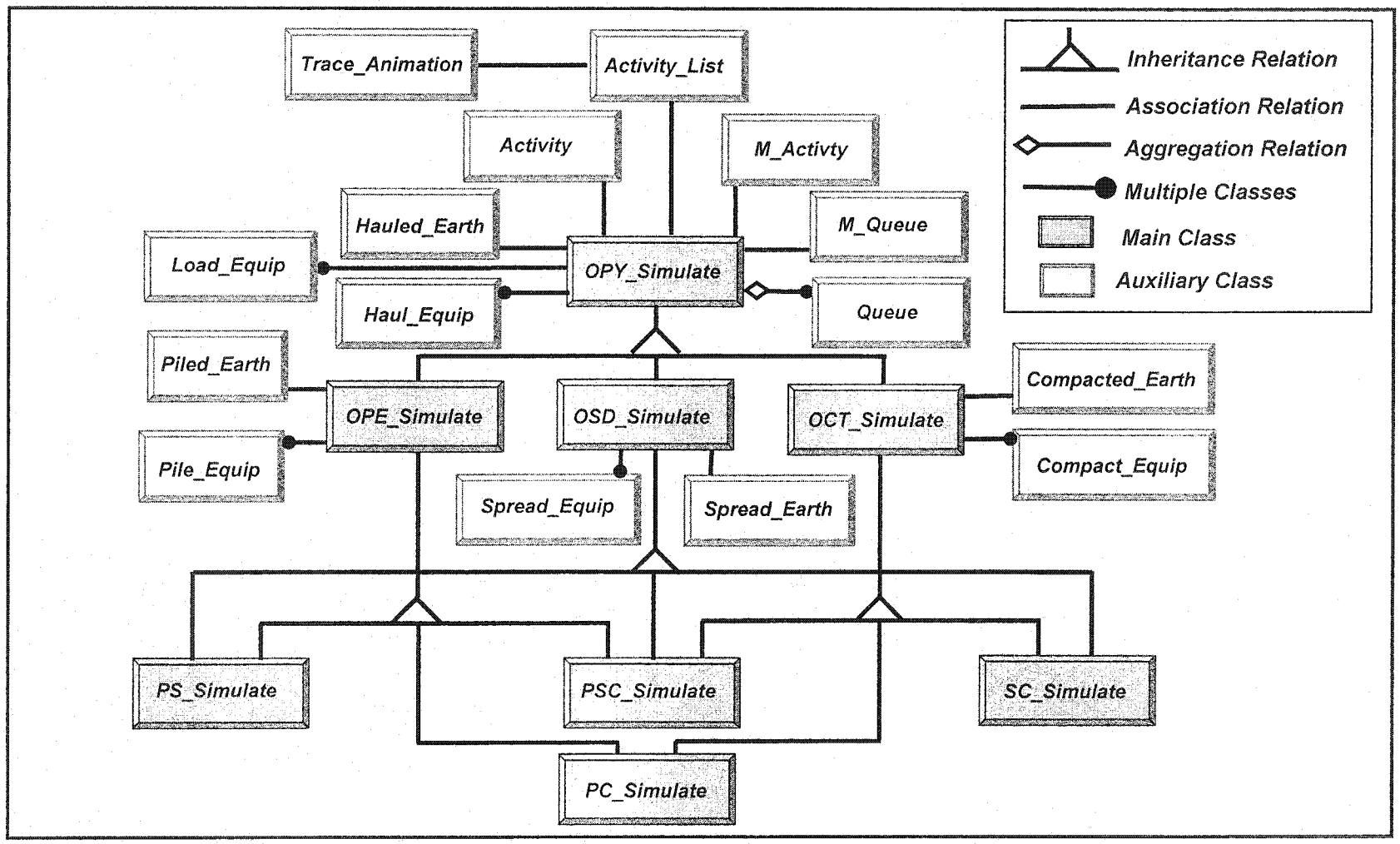


Figure 3.9 EMSP Main and Auxiliary Classes

Depending on the process to be simulated and its basic activities, *EMSP* automatically invokes the main class that represents the case and accordingly creates an object of that class. This is carried out through the `M_Simulate` class which has been designed to have an aggregate relationship with the rest of the main classes as shown in Figure 3.10. If interaction between equipment is not allowed, `M_Simulate` class creates objects from `SPT_Simulate` and `OPY_Simulate` classes to represent processes preformed by main and secondary activities, respectively (see Figure 3.11).

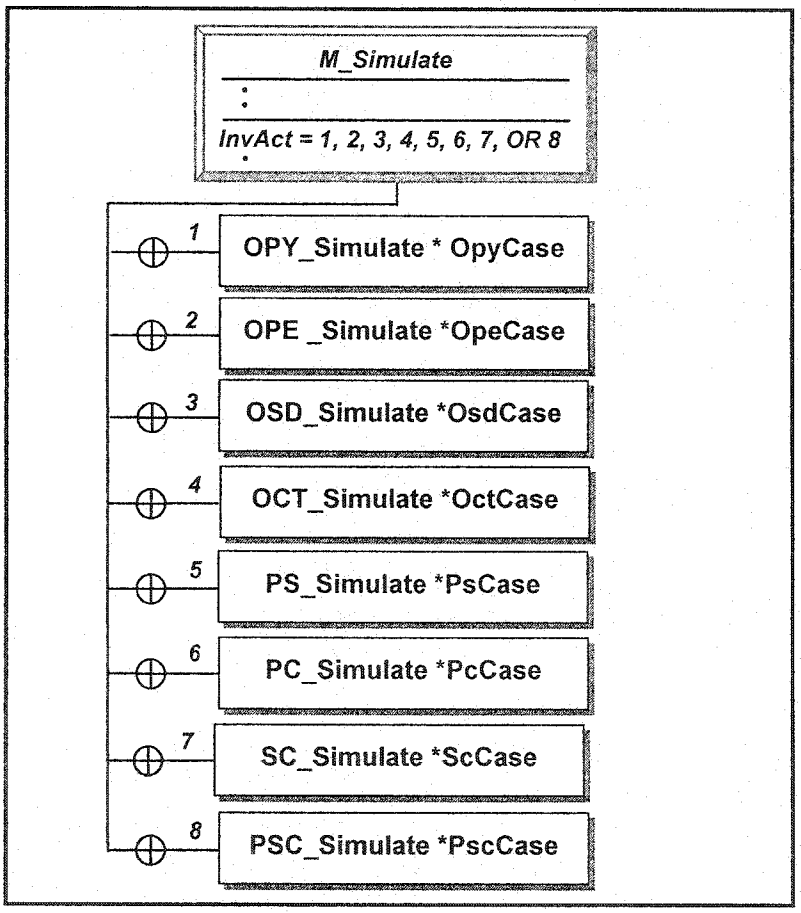


Figure 3.10 Object Creation for one of EMSP Main Classes

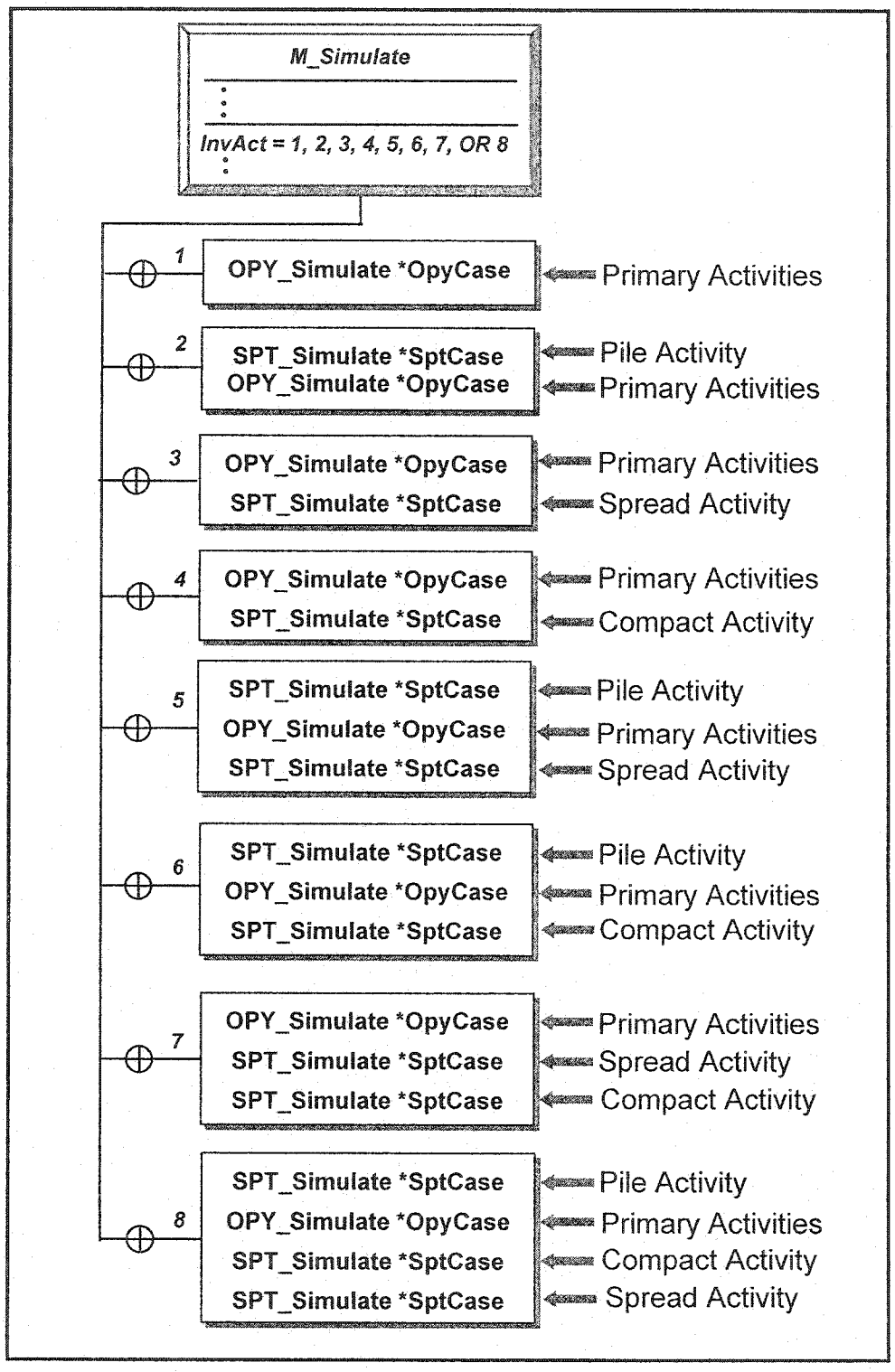


Figure 3.11 Creation of Objects (Interaction is Not Allowed)

3.4 EMSP Main Classes

EMSP classes are grouped into two categories: main and auxiliary. The main classes represent an earthmoving operation that contains any combination of main and secondary activities, allowing interaction between equipment. The eight main classes (`OPY_Simulate`, `OPE_Simulate`, `OSD_Simulate`, `OCT_Simulate`, `PS_Simulate`, `PC_Simulate`, `PSC_Simulate` and `SC_Simulate`) of the developed *EMSP* are described subsequently. The header files of these classes, which include member functions and data members, are included in Appendix A.

The `OPY_Simulate` class represents an earthmoving operation that consists of the four main activities: load, haul, dump and return (see Figure 3.12). It has been designed to act as a base class for the main classes used in *EMSP*, benefiting from the inheritance feature of object-orientation. It also has both association and aggregation relationships with the auxiliary classes.

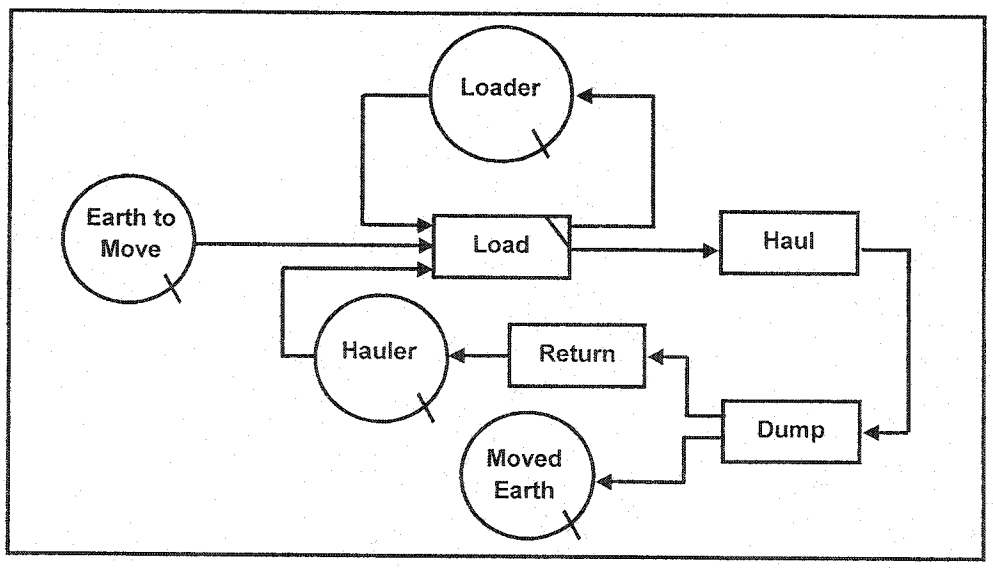


Figure 3.12 OPY_Simulate Using CYCLONE Notations

The association relationship is used with the `M_Queue`, `Activity`, `Activity_List`, `M_Activity`, `Hauled_Earth` and `Haul_Equip` classes, whereas, the aggregation relationship is used with the `Queue` class (see Figure 3.9). Creating an object of the `OPY_Simulate` class and invoking its member functions provide a complete replication of a simulation experiment. During simulation replication, the following tasks are carried out: 1) importing input data from external files; 2) creating objects of haulers (entities) and loaders (resources); 3) initializing simulation and setting entities and resources into their queues; 4) defining the probability distributions for the duration of the activities involved; 5) adding and removing objects from the current activity list (*CAL*); 6) checking the termination condition for the replication (e.g. all earth has been hauled); and 7) storing simulation statistics for further analysis.

Initiating simulation replication of an earthmoving operation that contains the four main activities, causes *EMSP* to allocate a memory space for an object of `OPY_Simulate` class. Seven member functions (principal functions) of that class will be called consecutively (see Figure 3.13):

1) `Define_Activities(..);`
2) `Initiate_Simulation();`
3) `Activity_Drive();`
4) `Store_QueueLength_Statistics(..);`
5) `Store_WaitTime_Statistics(..);`
6) `Store_Activities_Duration_Statistics(..);`
7) `Store_SecondHauler_Activities_Statistics(..).`

It should be noted that the last principal function is called only in the case of second hauler model existence.

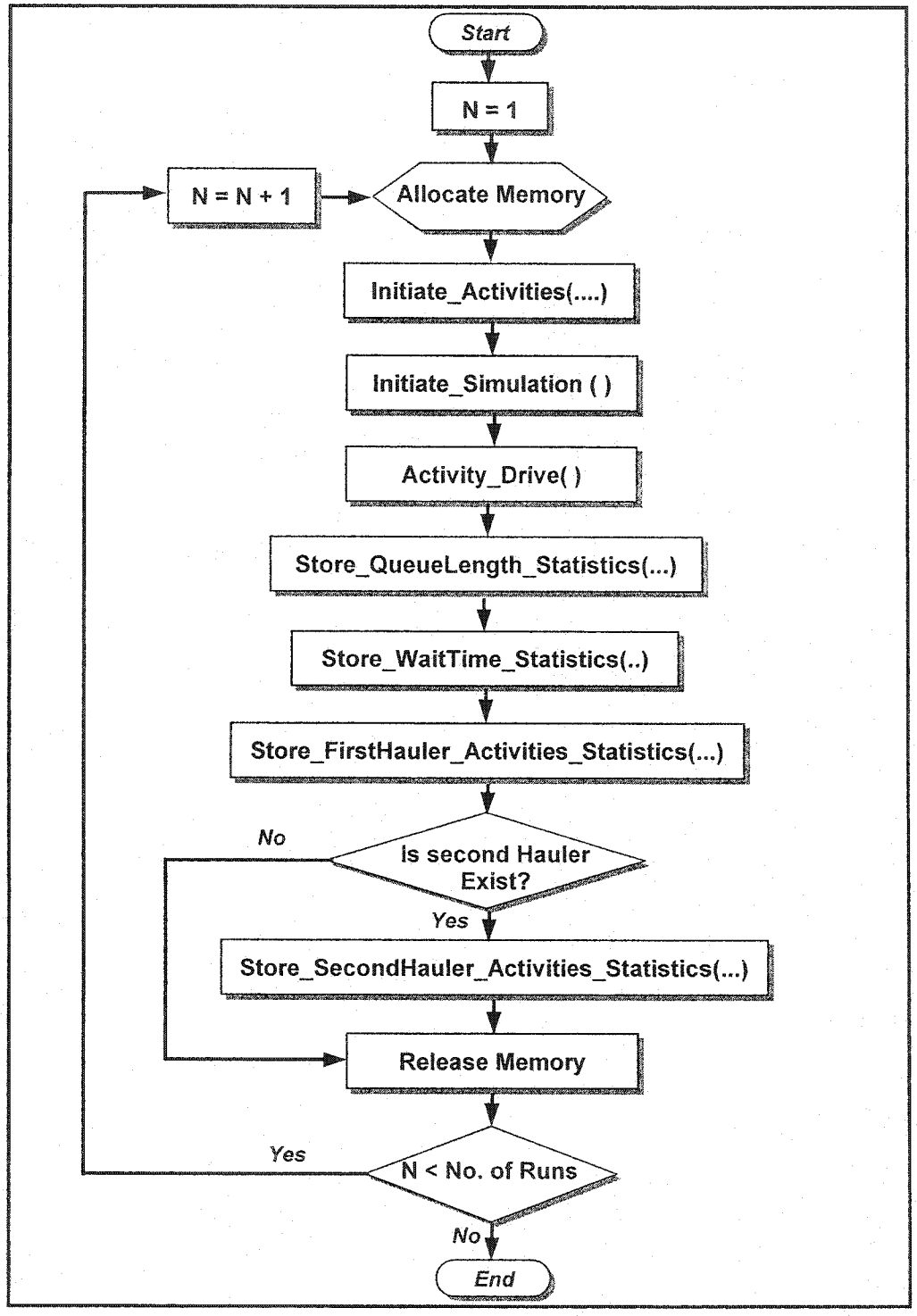


Figure 3.13 Flow of the Principal Functions

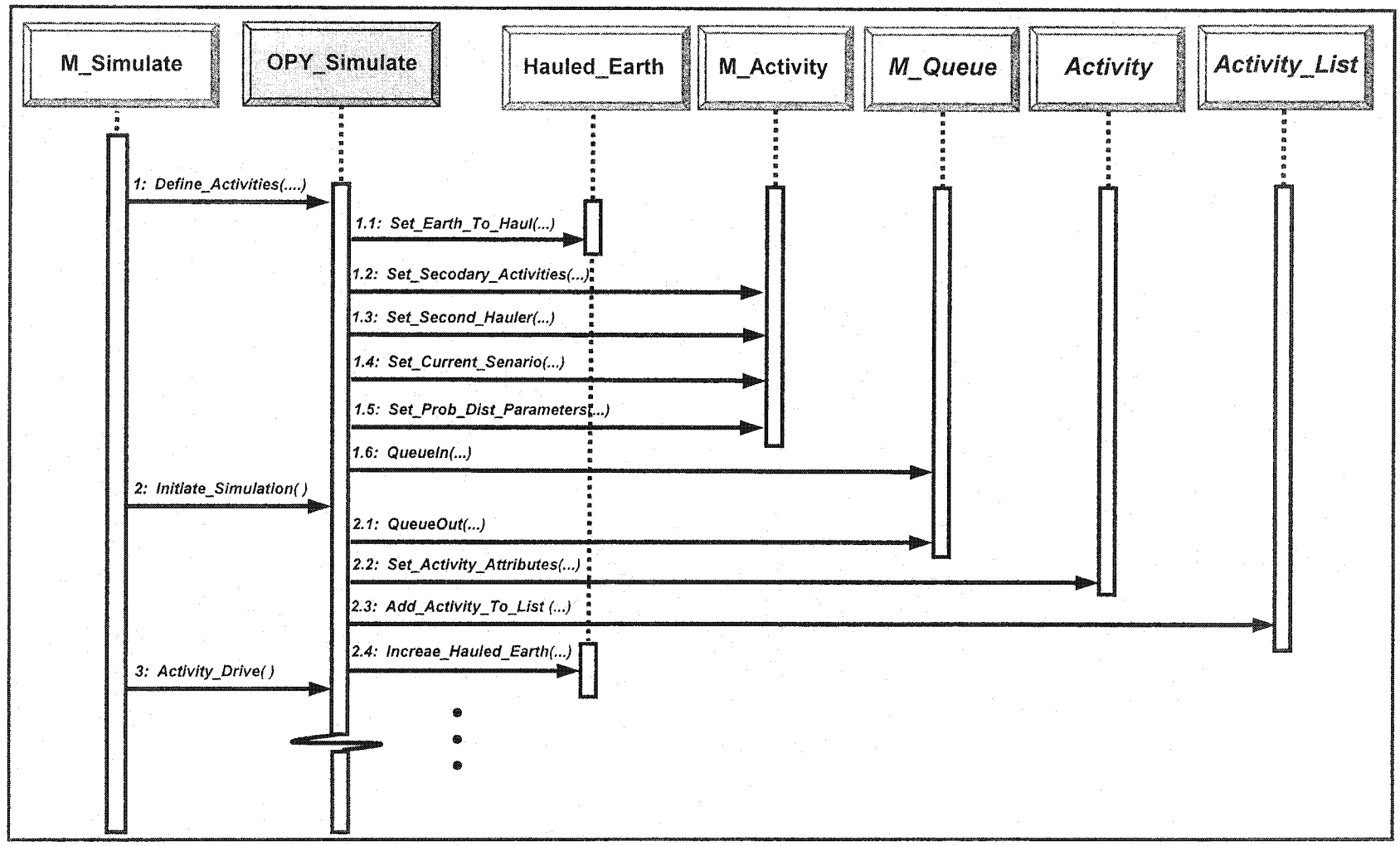


Figure 3.14 Sequence Diagram for Message Transfer from Principal Function

Each of these functions is responsible for performing different tasks whether by itself or by sending a message(s) to an object of a class that has an association or aggregation relationship with the `OPY_Simulate` class. Figure 3.14 depicts the sequence diagram that shows the progression of message sending to the other classes in order to perform a complete simulation run. On the other hand, the third principal function (`Activity_Drive()`) is responsible for calling four functions in the `OPY_Simulate` class. These functions are: 1) `Load_Drive();` 2) `Haul_Drive();` 3) `Dump_Drive();` and 4) `Return_Drive()`. They perform different tasks including: 1) adding and removing activities from CAL (current activity list); 2) checking termination condition for simulation replication; and 3) adding and removing haulers and loaders into their queues. Figure 3.15 illustrates how these functions are called within `Activity_Drive()`.

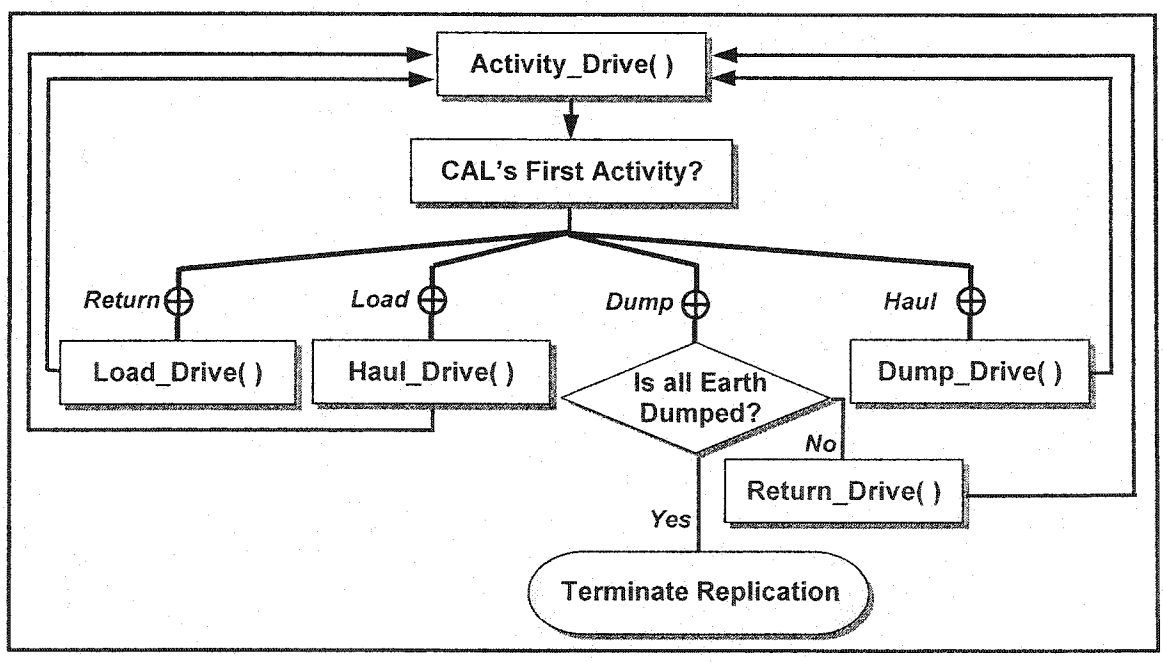


Figure 3.15 Calling OPY_Simulate Functions within Activity_Drive( )

The `OPE_Simulate` class represents an earthmoving operation that consists of pile activity in addition to the four main activities (see Figure 3.16). It is designed to be a subclass of the `OPY_Simulate` class. It has association relations with `Piled_Earth` and `Pile_Equip` classes in addition to all classes associated with the `OPY_Simulate` class (see Figure 3.9).

Creating an object of the `OPE_Simulate` class and invoking its member functions provide a complete replication of a simulation experiment. During simulation, pile equipment (resources) are created in addition to the stated tasks performed by `OPY_Simulate` objects, stated earlier. Initiating simulation replication of an earthmoving operation that contains pile and the four main activities, causes *EMSP* to allocate a memory space for an object of `OPE_Simulate` class and consequently to call the seven principal functions.

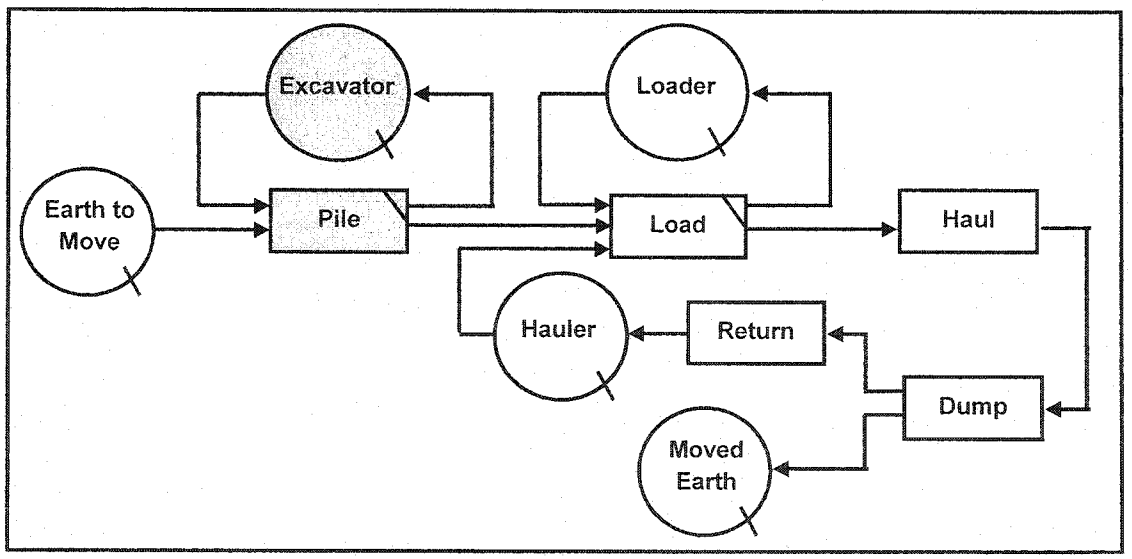


Figure 3.16 OPE_Simulate Using CYCLONE Notations

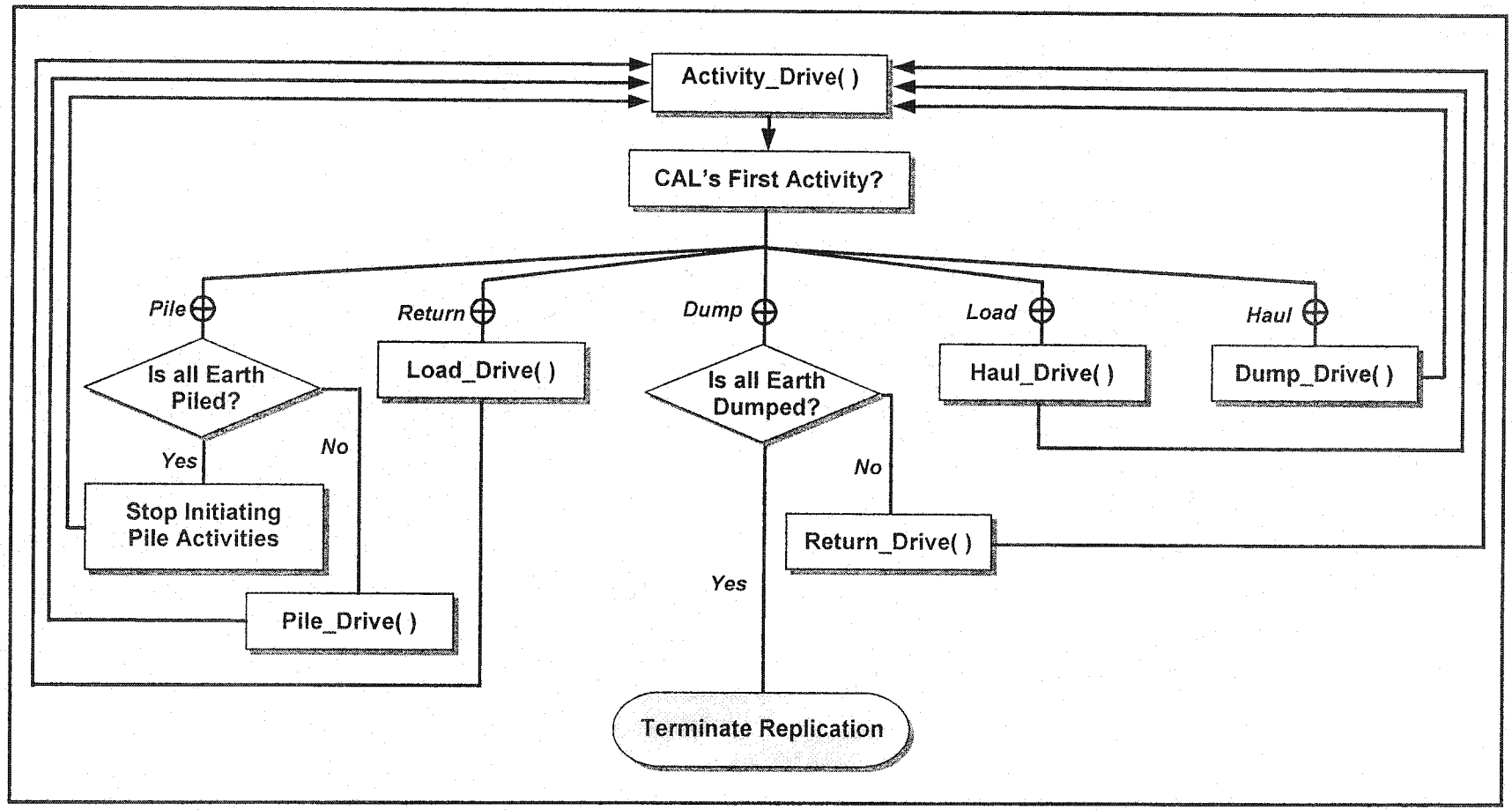


Figure 3.17 Calling OPE_Simulate Functions within Activity_Drive( )

The `Activity_Drive()` function is responsible for calling five functions, defined in the `OPE_Simulate` class. These functions are: 1) `Pile_Drive()`; 2) `Load_Drive()`; 3) `Haul_Drive()`; 4) `Dump_Drive()`; and 5) `Return_Drive()`. Figure 3.17 illustrates how these functions are called within `Activity_Drive()`.

The `OSD_Simulate` class represents an earthmoving operation that consists of spread activity in addition to the four main activities (see Figure 3.18). It is designed to be a subclass of the `OPY_Simulate` class. It has association relations with `Spread_Earth` and `Spread_Equip` classes in addition to all classes associated with the `OPY_Simulate` class (see Figure 3.9).

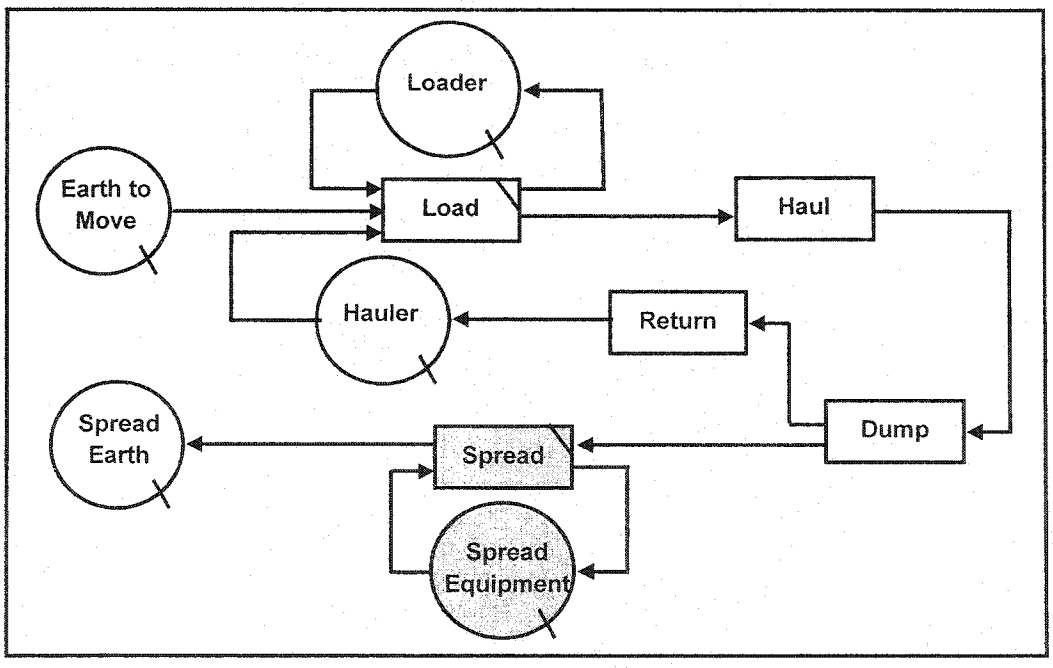


Figure 3.18 OSD_Simulate Using CYCLONE Notations

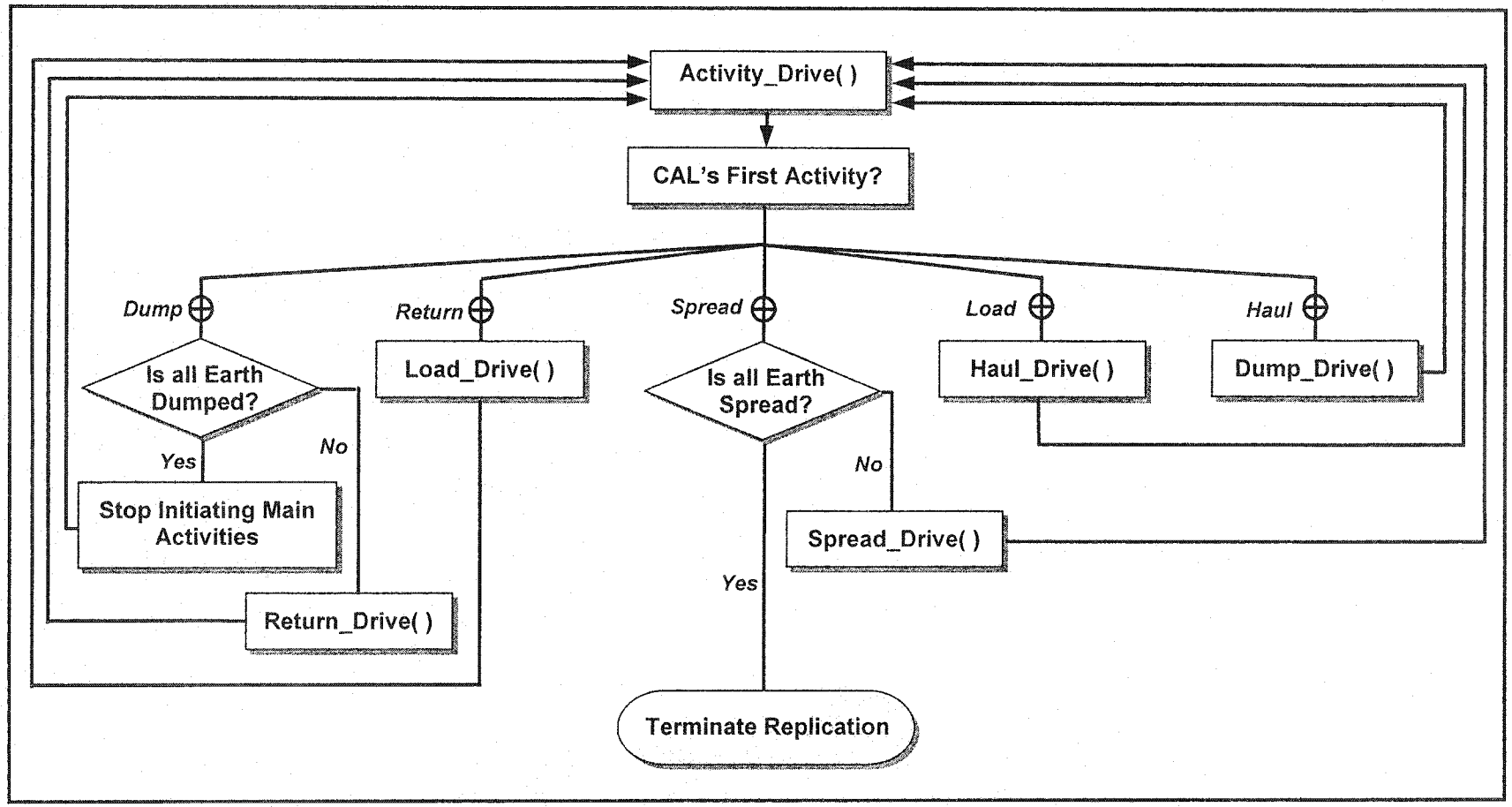


Figure 3.19 Calling OSD_Simulate Functions within Activity_Drive( )

Creating an object of the `OSD_Simulate` class and invoking its member functions provide a complete replication of a simulation experiment. During simulation, spread equipment (resources) are created in addition to the stated tasks performed by `OPY_Simulate` objects, stated earlier. Initiating simulation replication of an earthmoving operation that contains spread and the four main activities, causes *EMSP* to allocate a memory space for an object of `OSD_Simulate` class and consequently to call the seven principal functions. The `Activity_Drive()` function is responsible for calling five functions, defined in the `OSD_Simulate` class. These functions are: 1) `Load_Drive();` 2) `Haul_Drive();` 3) `Dump_Drive();` 4) `Return_Drive()` and 5) `Spread_Drive()`. Figure 3.19 illustrates how these functions are called within `Activity_Drive()`.

The `OCT_Simulate` class represents an earthmoving operation that consists of compact activity in addition to the four main activities (see Figure 3.20). It is designed to be a subclass of the `OPY_Simulate` class. It has association relations with `Compacted_Earth` and `Compact_Equip` classes in addition to all classes associated to `OPY_Simulate` class (see Figure 3.9). Creating an object of the `OCT_Simulate` class and invoking its member functions provide a complete replication of a simulation experiment. During simulation, compact equipment (resources) are created in addition to the stated tasks performed by `OPY_Simulate` objects, stated earlier. Initiating simulation replication of an

earthmoving operation that contains compact and the four main activities, causes *EMSP* to allocate a memory space for an object of `OCT_Simulate` class and consequently to call the seven principal functions.

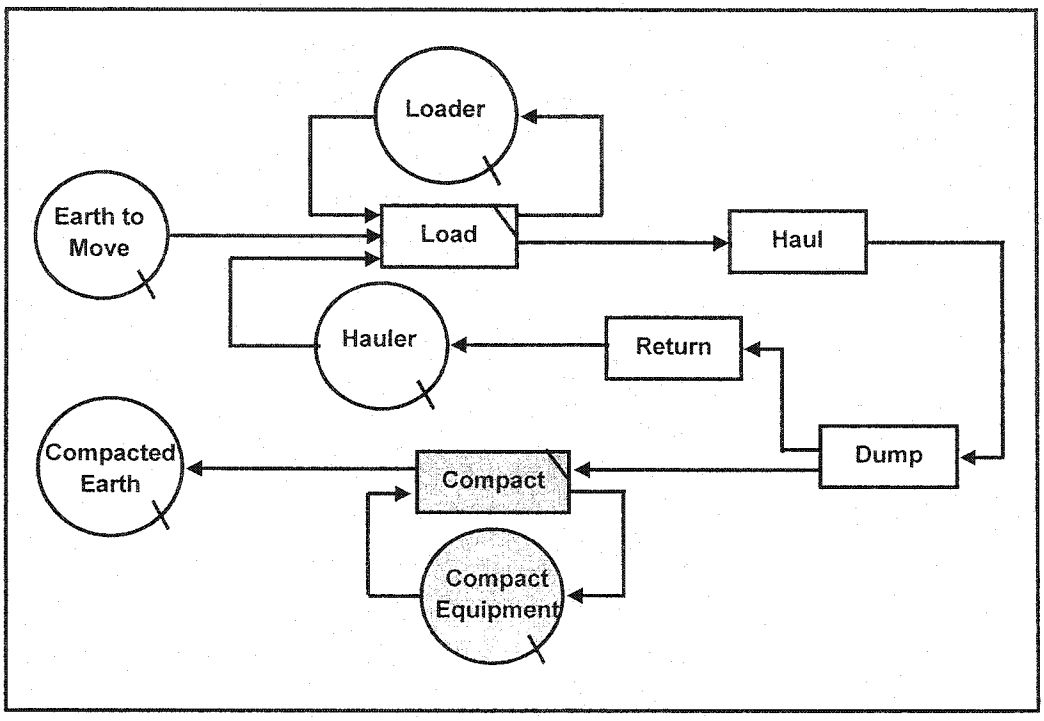


Figure 3.20 OCT_Simulate Using CYCLONE Notations

The `Activity_Drive()` function is responsible for calling five functions, defined in the `OCT_Simulate` class. These functions are: 1) `Load_Drive();` 2) `Haul_Drive();` 3) `Dump_Drive();` 4) `Return_Drive()` and 5) `Compact_Drive()`. Figure 3.21 illustrates how these functions are called within `Activity_Drive()`.

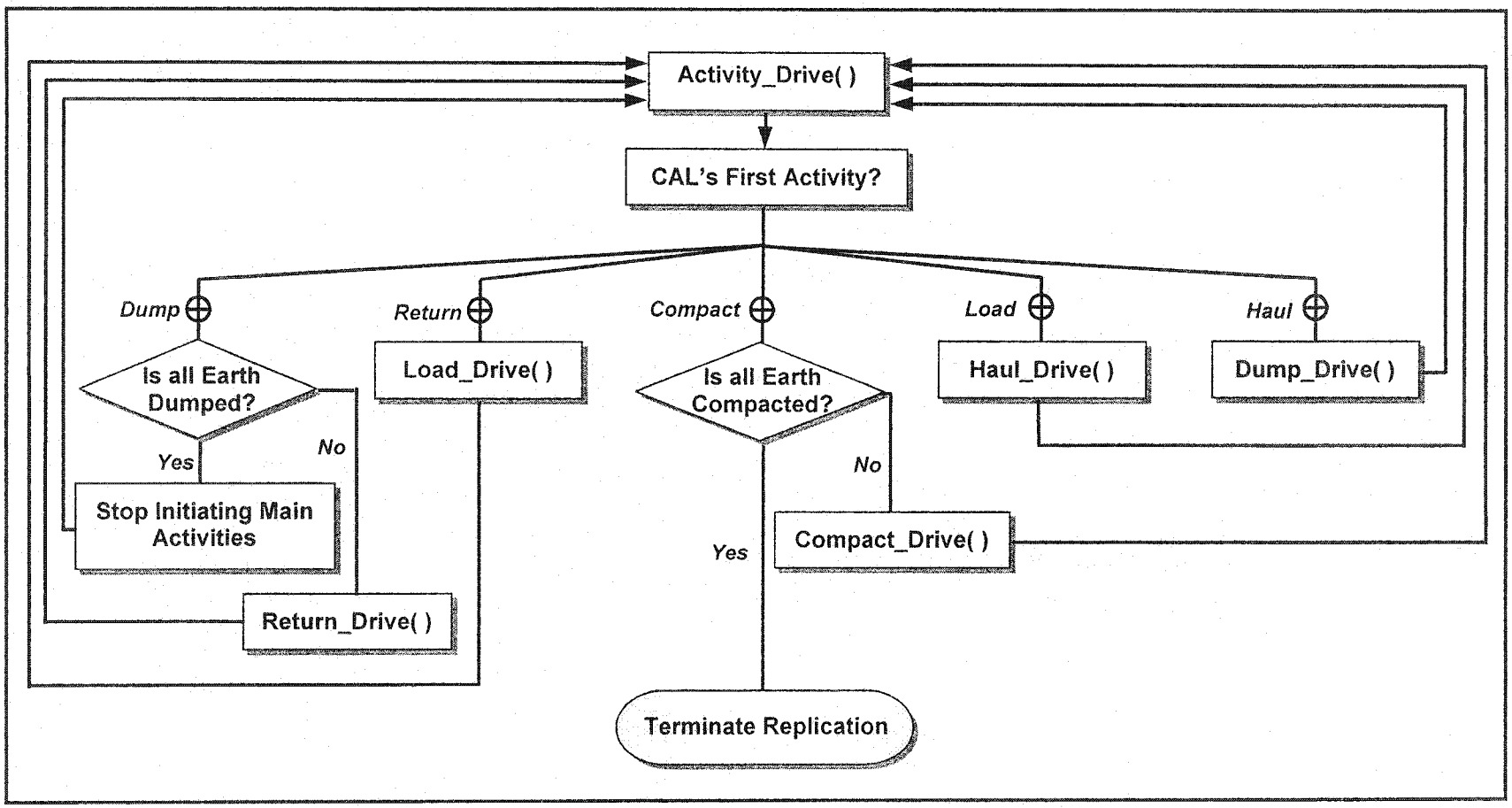


Figure 3.21 Calling OCT_Simulate Functions within Activity_Drive( )

The `PS_Simulate` class represents an earthmoving operation that consists of pile and spread activities in addition to the four main activities (see Figure 3.22). It is designed to be a subclass of both `OPE_Simulate` and `OSD_Simulate` classes. Creating an object of the `PS_Simulate` class and invoking its member functions provide a complete replication of a simulation experiment. During simulation, pile and spread equipment (resources) are created in addition to the stated tasks performed by `OPY_Simulate` objects, stated earlier. Initiating simulation replication of an earthmoving operation that contains pile, spread and the four main activities, causes *EMSP* to allocate a memory space for an object of `PS_Simulate` class and consequently to call the seven principal functions.

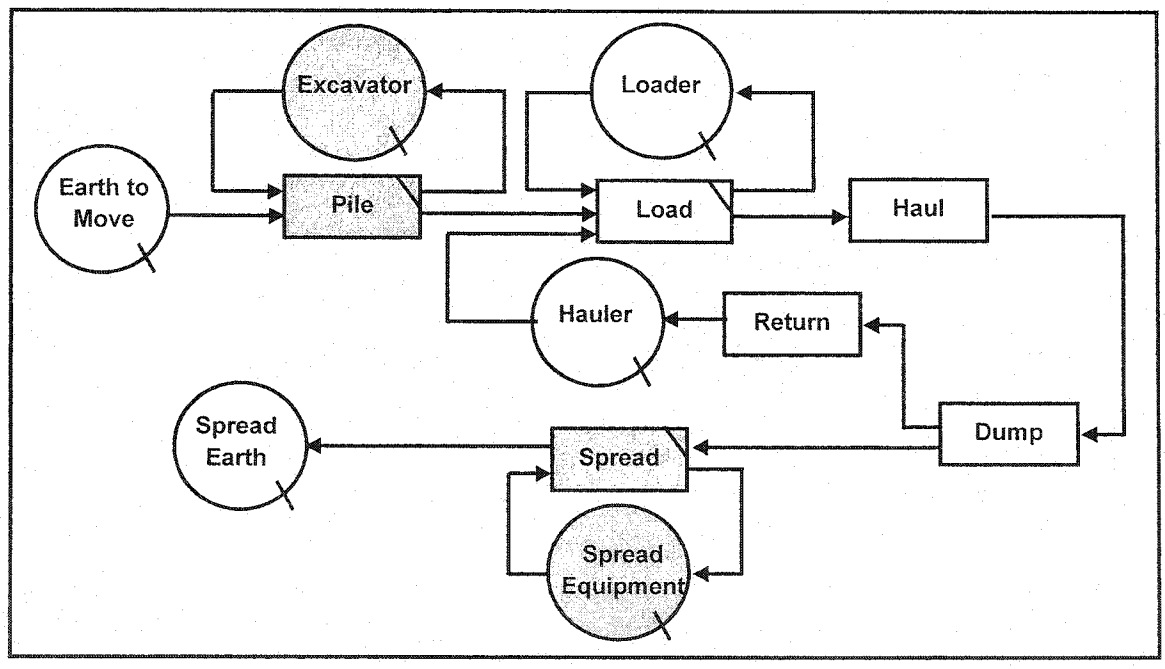


Figure 3.22 PS_Simulate Using CYCLONE Notations

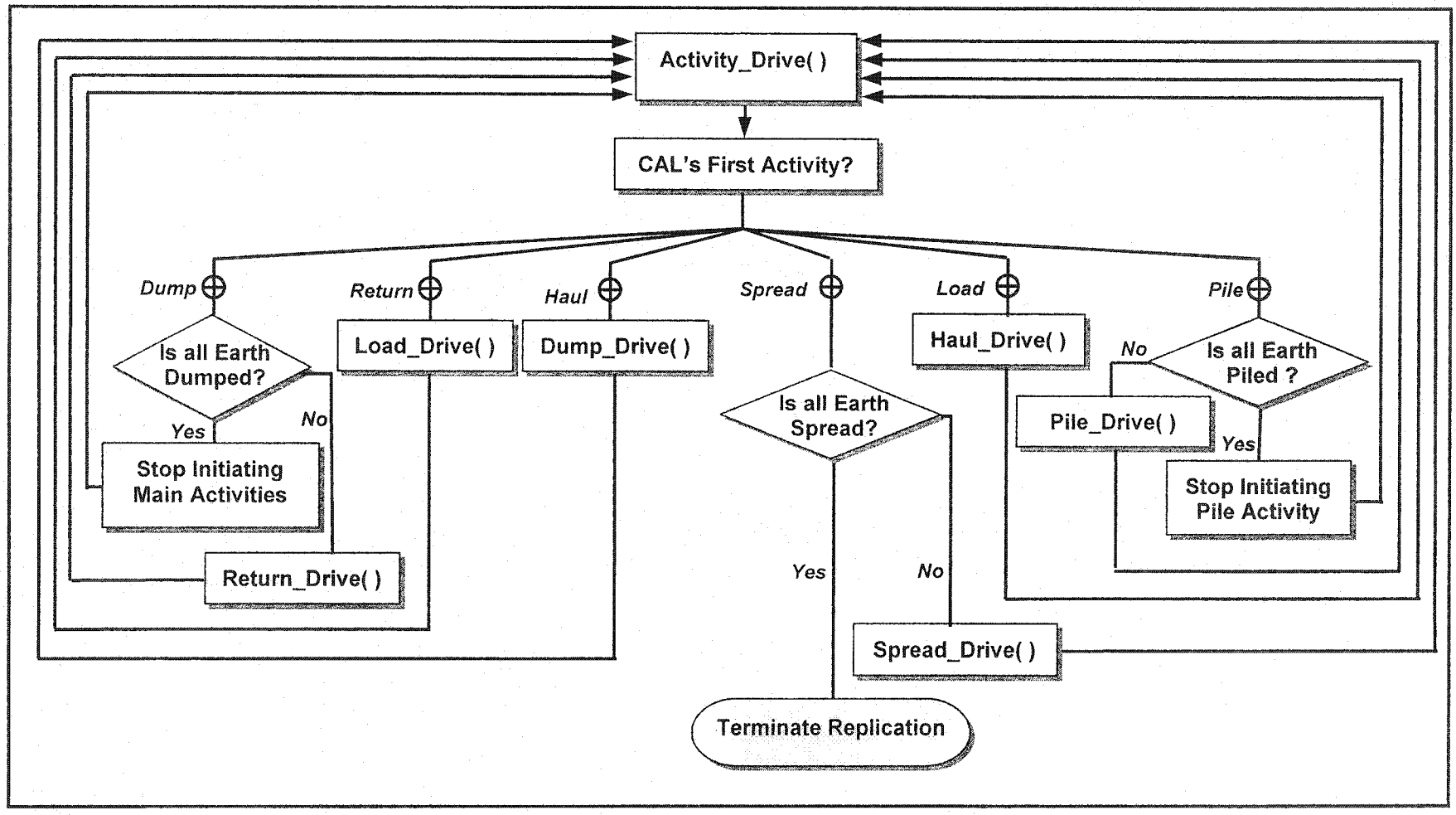


Figure 3.23 Calling PS_Simulate Functions within Activity_Drive( )

The `Activity_Drive()` function is responsible for calling six functions, defined in the `PS_Simulate` class. These functions are: 1) `Pile_Drive()`; 2) `Load_Drive()`; 3) `Haul_Drive()`; 4) `Dump_Drive()`; 5) `Return_Drive()` and 6) `Spread_Drive()`. Figure 3.23 illustrates how these functions are called within `Activity_Drive()`.

The `PC_Simulate` class represents an earthmoving operation that consists of pile and compact activities in addition to the four main activities (see Figure 3.24). It is designed to be a subclass of both `OPE_Simulate` and `OCT_Simulate` classes. Creating an object of the `PC_Simulate` class and invoking its member functions provide a complete replication of a simulation experiment.

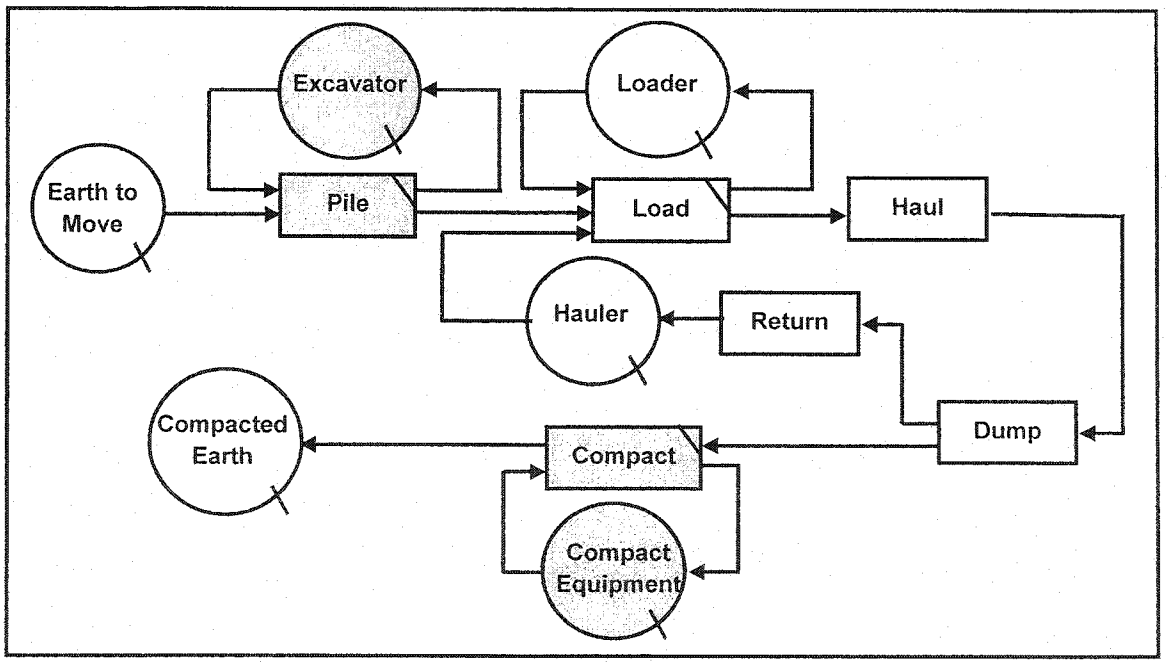


Figure 3.24 PC_Simulate Using CYCLONE Notations

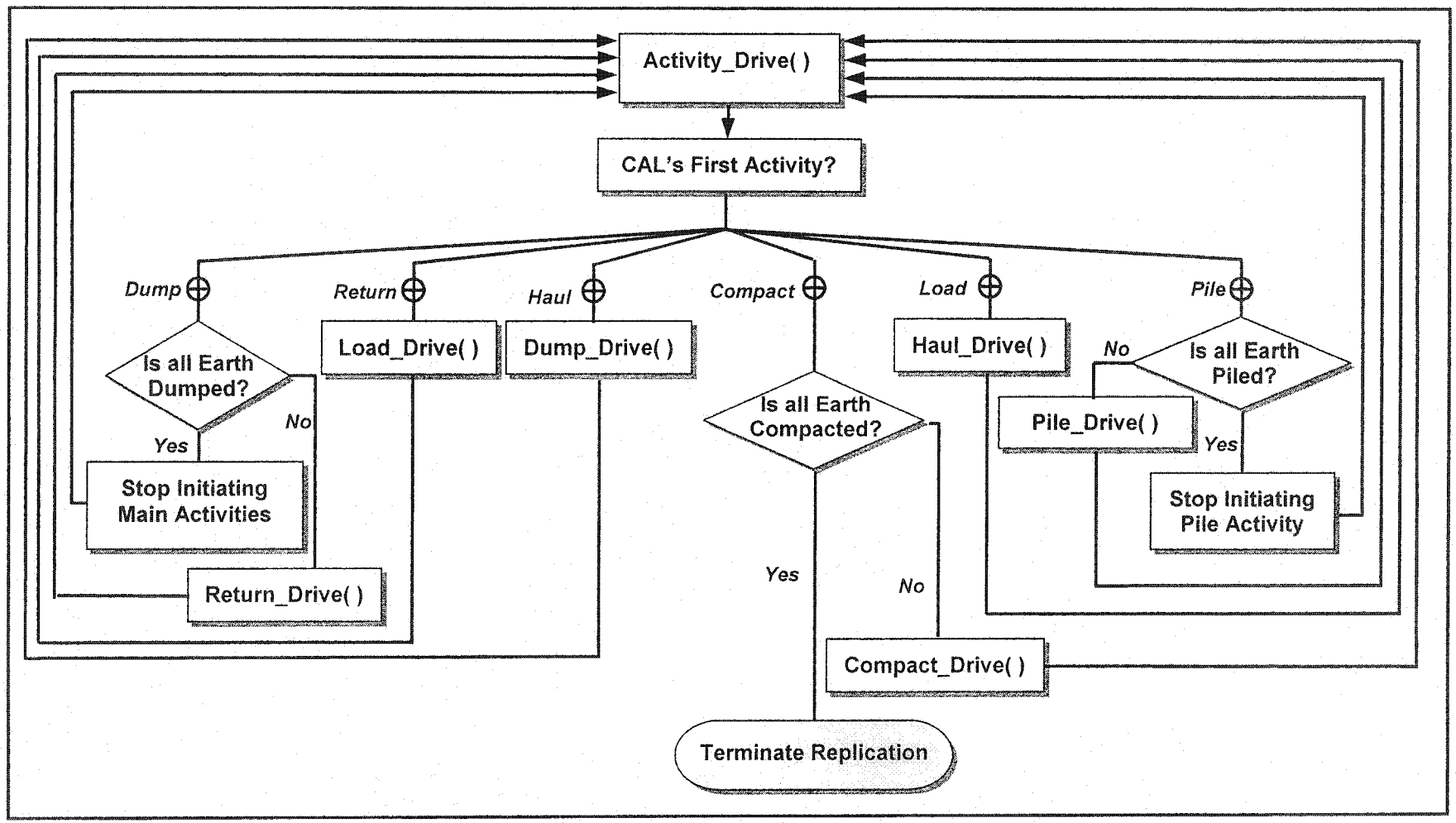


Figure 3.25 Calling PC_Simulate Functions within Activity_Drive( )

During simulation, pile and compact equipment (resources) are created in addition to the stated tasks performed by `OPY_Simulate` objects, stated earlier. Initiating simulation replication of an earthmoving operation that contains pile, compact and the four main activities, causes *EMSP* to allocate a memory space for an object of `PC_Simulate` class and consequently to call the seven principal functions. The `Activity_Drive()` function is responsible for calling six functions, defined in the `PC_Simulate` class. These functions are: 1) `Pile_Drive()`; 2) `Load_Drive()`; 3) `Haul_Drive()`; 4) `Dump_Drive()`; 5) `Return_Drive()` and 6) `Compact_Drive()`. Figure 3.25 illustrates how these functions are called within `Activity_Drive()`.

The `SC_Simulate` class represents an earthmoving operation that consists of spread and compact activities in addition to the four main activities (see Figure 3.26). It is designed to be a subclass of both `OSD_Simulate` and `OCT_Simulate` classes. Creating an object of the `SC_Simulate` class and invoking its member functions provide a complete replication of a simulation experiment. During simulation, spread and compact equipment (resources) are created in addition to the stated tasks performed by `OPY_Simulate` objects, stated earlier. Initiating simulation replication of an earthmoving operation that contains spread, compact and the four main activities, causes *EMSP* to allocate a memory space for an object of `SC_Simulate` class and consequently to call the seven principal functions. The `Activity_Drive()` function is responsible

for calling six functions, defined in the `SC_Simulate` class. These functions are: 1) `Load_Drive();` 2) `Haul_Drive();` 3) `Dump_Drive();` 4) `Return_Drive();` 5) `Spread_Drive()` and 6) `Compact_Drive()`. Figure 3.27 illustrates how these functions are called within `Activity_Drive()`.

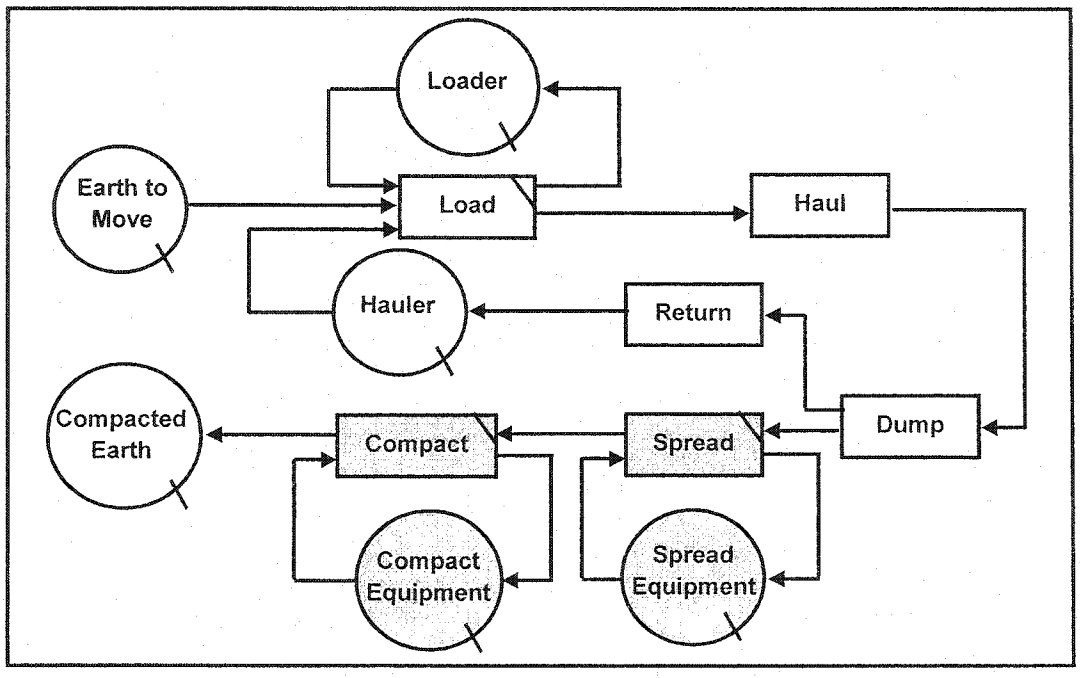


Figure 3.26 SC_Simulate Using CYCLONE Notations

The `PSC_Simulate` class represents an earthmoving operation that consists of pile, spread and compact activities in addition to the four main activities (see Figure 3.28). It is designed to be a subclass of `OPE_Simulate`, `OSD_Simulate` and `OCT_Simulate` classes. Creating an object of the `PSC_Simulate` class and invoking its member functions provide a complete replication of a simulation experiment.

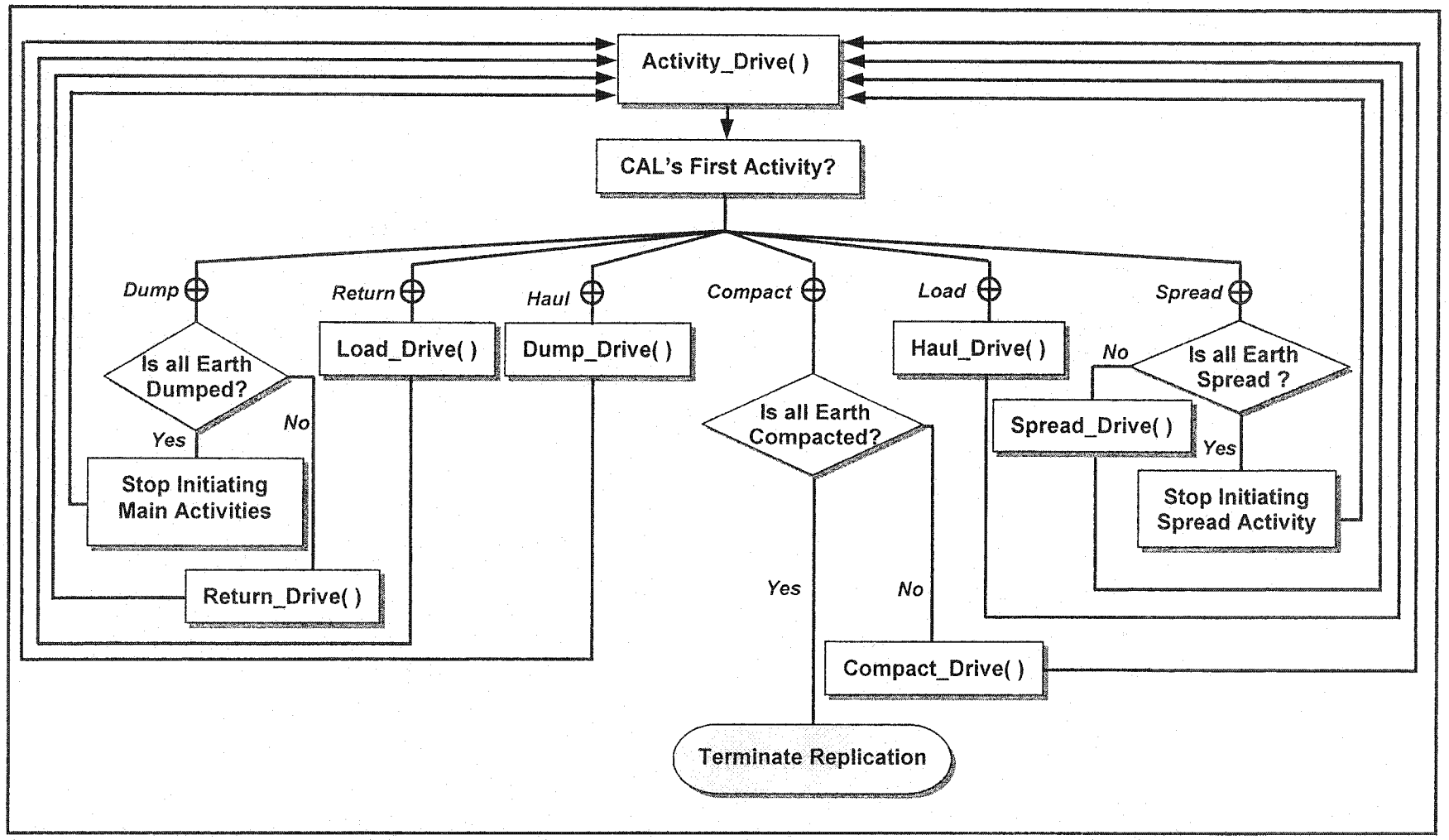


Figure 3.27 Calling SC_Simulate Functions within Activity_Drive()

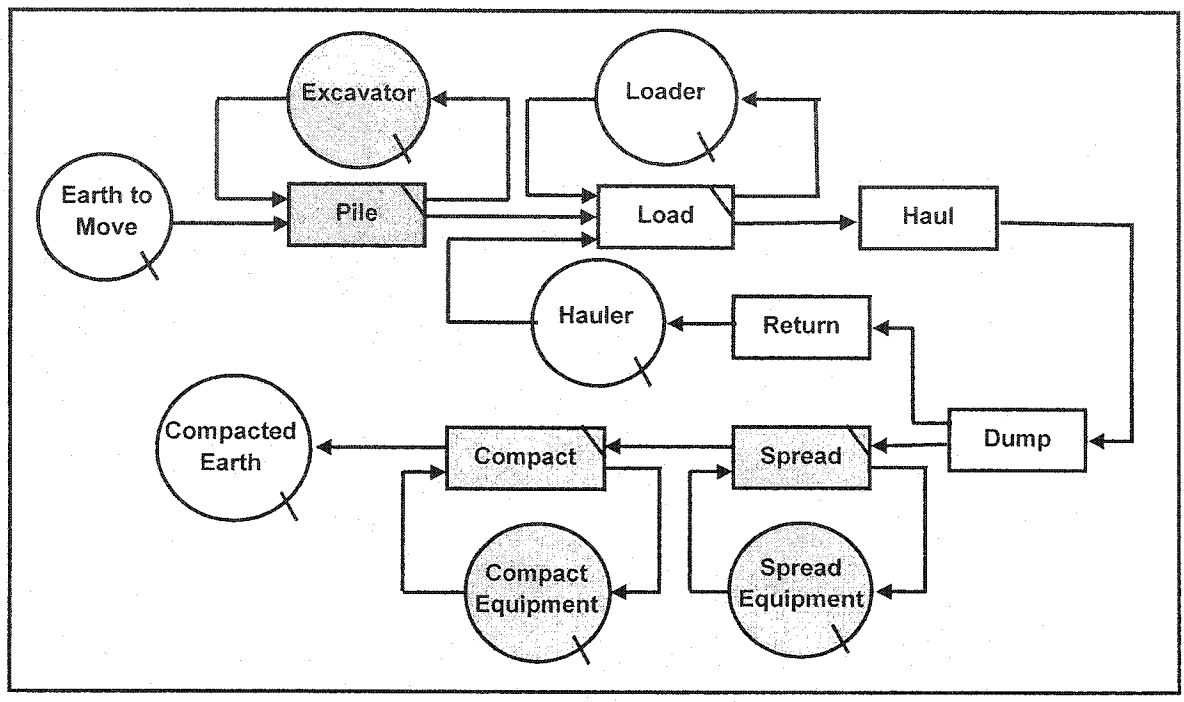


Figure 3.28 PSC_Simulate Using CYCLONE Notations

During simulation, pile, spread and compact equipment (resources) are created in addition to the stated tasks performed by `OPY_Simulate` objects, stated earlier. Initiating simulation replication of an earthmoving operation that contains pile, spread, compact and the four main activities, causes *EMSP* to allocate a memory space for an object of `PSC_Simulate` class and consequently to call the seven principal functions. The `Activity_Drive()` function is responsible for calling seven functions, defined in the `PSC_Simulate` class. These functions are: 1) `Pile_Drive();` 2) `Load_Drive();` 3) `Haul_Drive();` 4) `Dump_Drive();` 5) `Return_Drive();` 6) `Spread_Drive()` and 7) `Compact_Drive()`. Figure 3.29 illustrates how these functions are called within `Activity_Drive()`.

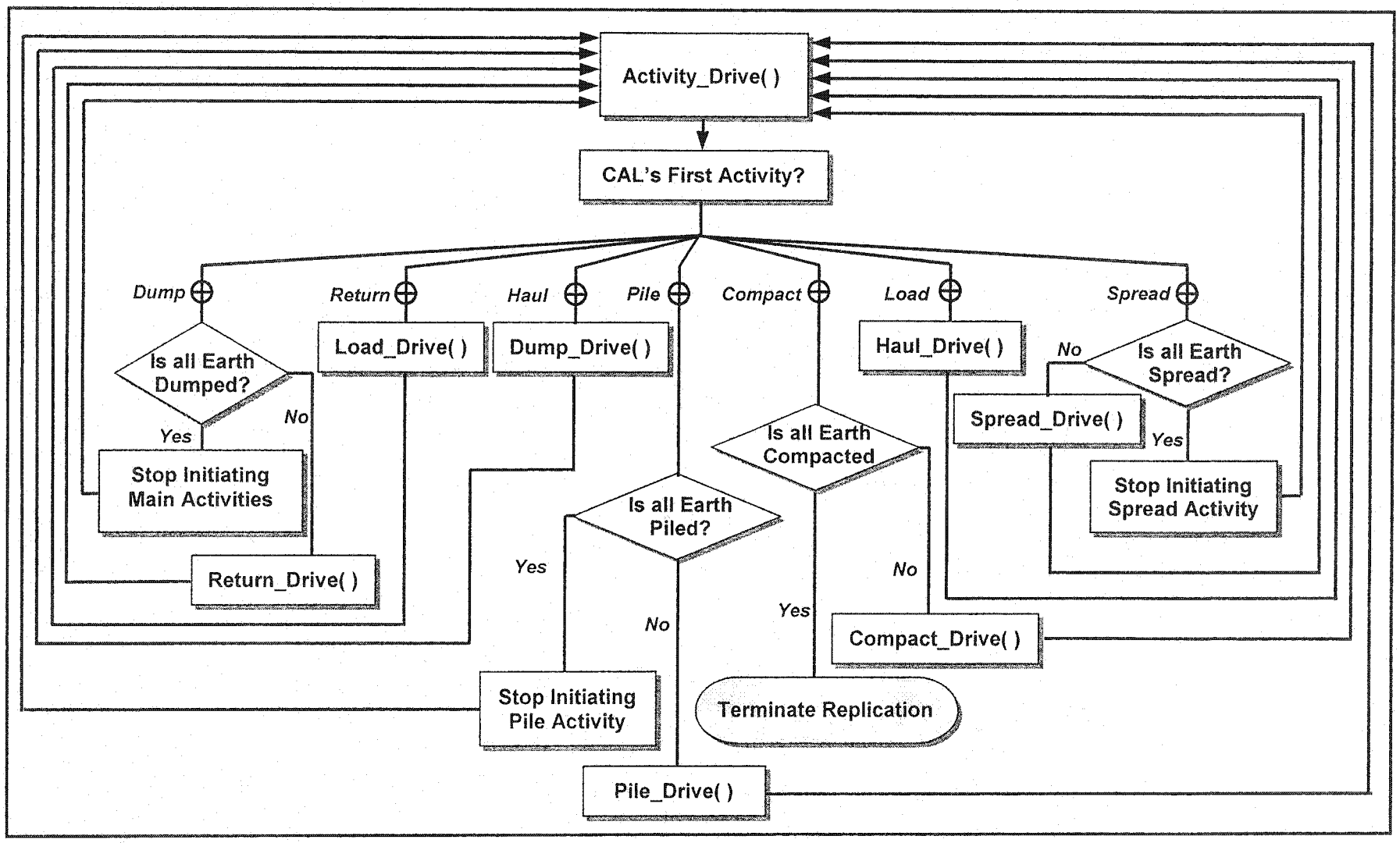


Figure 3.29 Calling PSC_Simulate Functions within Activity_Drive( )

3.5 EMSP Auxiliary Classes

Auxiliary classes are designed to support a wide range of functions carried out from within the main classes described earlier. They primarily provide: 1) aggregation and association relations with the main classes; 2) tracking capabilities of entities and resources during simulation; and 3) tracking of interim calculations gathered during simulation and other procedure computations. The auxiliary classes (`SPT_Simulate`, `M_Simulate`, `Activity`, `Activity_List`, `Queue`, `M_Queue`, `Equipment`, `Piled_Earth`, `Hauled_Earth`, `Spread_Earth`, `Compacted_Earth`, `Trace_Animation`) of the developed *EMSP* are described subsequently. The header files of these classes, which include member functions and data members, are included in Appendix A.

The `SPT_Simulate` class is used to represent a process performed by support equipment (e.g. pile, spread and compact). It is jointly utilized with the `OPY_Simulate` class to represent an earthmoving operation that does not allow for interaction between main activities' equipment and secondary activities' equipment (see Figure 3.11). Initiating simulation for an earthmoving operation and allocating a memory space for an object(s) of `SPT_Simulate` class cause *EMSP* to call the following six principal functions;

```
1) Define_Activities(..);
```

```
2) Initiate_Simulation();
3) Activity_Drive();
4) Store_QueueLength_Statistics(..);
5) Store_WaitTime_Statistics(..);
6) Store_Activities_Duration_Statistics(..);
```

The `Activity_Drive()` function is responsible for recursively calling the `Support_Drive()` function.

The `M_Simulate` class is designed to allocate memory space and to call the principal functions associated with the eight main classes one at a time. Where interaction between secondary and main activities is not allowed, the `M_Simulate` class carries out the principal functions for `OPY_Simulate` and `SPT_Simulate` classes only. The allocation of memory and calling the principal functions are performed by `Simulate_Operations()` function, defined as a member function in the `M_Simulate` class. The `Simulate_Operations()` function is also responsible for exporting simulation outputs in order to allow the *ORM* static sub-module to generate its reports by triggering three functions defined in `M_Simulate` class.

The `Activity` class is designed to represent the characteristics of all involved activities (e.g. pile, load, haul, dump, return, spread and compact). These characteristics include: 1) activity identification, 2) activity start time, 3) activity duration, 4) activity finish time and 5) ID of equipment used in the activity.

These characteristics are assigned to an activity utilizing `Set_Activity_Attributes(.)` function, defined in `Activity` class.

The `Activity_List` class is an application of dynamic data structure where data are collected from objects (created from `Activity` class) without knowing their number in advance. It is a singly linked list, which is defined with the assistance of a supporting class called `AcElement`. The `Activity_List` class is designed to perform two main tasks: 1) adding and removing activities from CAL (current activity list) and 2) triggering different functions defined in the `Trace_Animation` class. The first task is carried out by `Add_Activity_To_List (..)` and `Remove_Activity_From_List()` functions, respectively. The second task is carried out by both `Trace_Add_Manager(..)` and `Trace_Remove_Manager(..)` functions.

The `Queue` class is another application of dynamic data structure, defined in *EMSP* utilizing "*Template*" terminology. This terminology allows the `Queue` class to handle with different types of objects (e.g. haul equipment, load equipment, etc.). Objects created from `Queue` class allow adding and removing queues' objects from the queue according to First in, First out (i.e. objects are added at the end of the queue and removed from the beginning). The `Queue` class is also a singly linked list, which is defined with the assistance of a support class called `Element`. Objects are added and removed from `Queue` class utilizing `EnQueue(..)` and `DeQueue()` functions, respectively.

The M_Queue is a control class that does not have data members. It is designed to enhance the functionality of objects created from the Queue class. This enhancement is carried out through several member functions that receive the name of the Queue object as an argument. These functions are responsible for: 1) inserting objects at the beginning of a queue using QueueIn(..) function; 2) retrieving queue length using Get_Length(..) function; 3) retrieving queue's first element using Get_First_Element(..) function; 4) removing element from the end of a queue using QueueOut(..) function; and 5) checking whether a queue is empty or not using IsQueueEmpty(..) function.

The Equipment class is designed to represent the characteristics of earthmoving equipment. Objects created from that class, store equipment identification (ID) and the time when they are inserted into their respective queue using Equip_No and Entrance_Time data members of this class, respectively. Five subclasses (Haul_Equip, Load_Equip, Pile_Equip, Spread_Equip and Compact_Equip) are designed to inherit the characteristics of the base class (Equipment). For example, the Haul_Equip class has two data members (Hauler_Type and PayLoad) which add more specialization to haul equipment. These data members capture the type of the hauler model and its payload, respectively.

The Piled_Earth class is designed to represent the quantity of earth to be piled and its change during simulation time. Only one object is created from that

class in a simulation replication. It exists only in simulation experiments that contain pile activity. Triggering a member function of the `Piled_Earth` class performs one of the following tasks: 1) inputting the quantity of earth to be piled; 2) calculating the accumulative quantity of piled earth; 3) calculating the quantity of earth yet to be piled; 4) checking if the amount of the piled earth is enough to load a hauler and 5) checking if the piling process is completed or not.

The `Hauled_Earth` class is designed to represent the quantity of earth to be hauled and its change during simulation time. Only one object is created from that class in a simulation replication. This object should exist in all simulation experiments that have the four main activities. Triggering a member function of the `Hauled_Earth` class performs one of the following tasks: 1) inputting the quantity of earth to be hauled; 2) calculating the accumulative quantity of hauled earth; 3) calculating the quantity of earth yet to be spread or compacted (if these activities are to exist); 4) checking the possibility to perform spread or compact activities (if they are to exist) and 5) checking if the hauling process is completed or not.

The `Spread_Earth` class is designed to represent the quantity of earth to be spread and its change during simulation time. Only one object is created from that class in a simulation replication. It exists only in simulation experiments that contain spread activity. Triggering a member function of the `Spread_Earth` class performs one of the following tasks: 1) inputting the quantity of earth to be

spread; 2) calculating the accumulative quantity of earth that has been spread; 3) calculating the quantity of earth yet to be compacted (if compact activity is to exist); 4) checking the possibility to perform compact activity (if it is to exist) and 5) checking if the spreading process is completed or not.

The `Compacted_Earth` class is designed to represent the quantity of earth to be compacted and its change during simulation time. Only one object is created from that class in a simulation replication. It exists only in simulation experiments that contain compact activity. Triggering a member function of the `Compacted_Earth` class performs one of the following tasks: 1) inputting the quantity of earth to be spread; 2) calculating the accumulative quantity of compacted earth; and 3) checking if the compaction process is completed or not.

The `Trace_Animation` class is designed to prepare a suitable input file for the Proof Animation software (2000). It is invoked from within the dynamic sub-module of *ORM*. The prepared input file contains: 1) the location of the resources and entities throughout the experiment time, and 2) the duration that has been utilized by those resources and entities at the different location. That file is written in a special code compatible with the Proof Animation software.

3.6 Numerical Example

A case study of a quarry has been considered in order to validate *EMSP* and demonstrate its essential capabilities. The case simply involved hauling excavated rock from a quarry area to the location of a crusher. The data of the case are summarized in Table 3.5. It is required to select an optimum fleet configuration from a list of equipment available to the contractor so as to yield minimum project duration. To enable a comparison, the case has been analyzed using the developed simulation engine (*EMSP*) and the commercial software *FPC* (1998). Figure 3.30 provides a layout of the haul road that links the quarry area to the location of the crusher. An alternate haul road (see Figure 3.31), crossing a private property, which could be used for a fee is to be considered in the analysis so as to examine its viability. The differences of the characteristics of the two haul roads are highlighted as shown in Figures 3.30 and 3.31. The characteristics of haul road segments have been verified to ensure to ensure consistency and accuracy. For example, the elevation of the end point for the alternative road (Segment 5 in Figure 3.31) should be identical to the elevation of the end point for segment 10 (Figure 3.30) in the original road segment. It should be noted that all segment grades have been reviewed to satisfy that condition.

Four different fleet scenarios have been generated form the different combinations of equipment and haul roads, see Table 3.6. The analysis has been performed utilizing *FPC* and the input data listed in Table 3.7. The analysis has also been performed utilizing *EMSP*. The probability density functions for the

durations of activities, utilized by EMSP, are listed in Table 3.8 and a 95% job efficiency is considered in the analysis.

Table 3.5 Data for the Numerical Example

Soil Material Properties	
Scope of Work:	225,000 ton
Bank Density:	1.93 t/m^3
Loose Density:	1.72 t/m^3
Load Factor:	89 %
Road Rolling Resistance:	3%
Load Equipment Characteristics	
Type:	Wheel Type
Model:	988F II
Number:	1
Bucket Capacity:	6.12 m^3
Hourly Owing & Operating Cost:	135.00 ($/hr.)
Haul Equipment Characteristics	
Type:	Off-Highway Trucks
Model:	769D
Number:	2
Max. Capacity:	36.8 ton
Hourly Owing & Operating Cost:	110.00 ($/hr.)
Type:	Off-Highway Trucks
Model:	771D
Number:	2
Max. Capacity:	40.0 ton
Hourly Owing & Operating Cost:	114.00 ($/hr.)

Table 3.6 Fleet Scenarios

	Fleet_1 *(One 988F II + Two 769D)*	**Fleet_2** *(One 988F II + Two 771D)*
Current Haul Road	**Scenario_1**	**Scenario_2**
Alternative Haul Road	**Scenario_3**	**Scenario_4**

Original Haul Road Segments

Segment	Length (m)	Grade (%)
1	20	0
2	3	0
3	125	6
4	110	0
5	160	-8
6	190	0
7	100	-3
8	360	1
9	160	-12
10	140	1
11	100	-9
12	185	0
13	25	7
14	99	10
15	26	1
16	3	0
17	18	0
	1824	

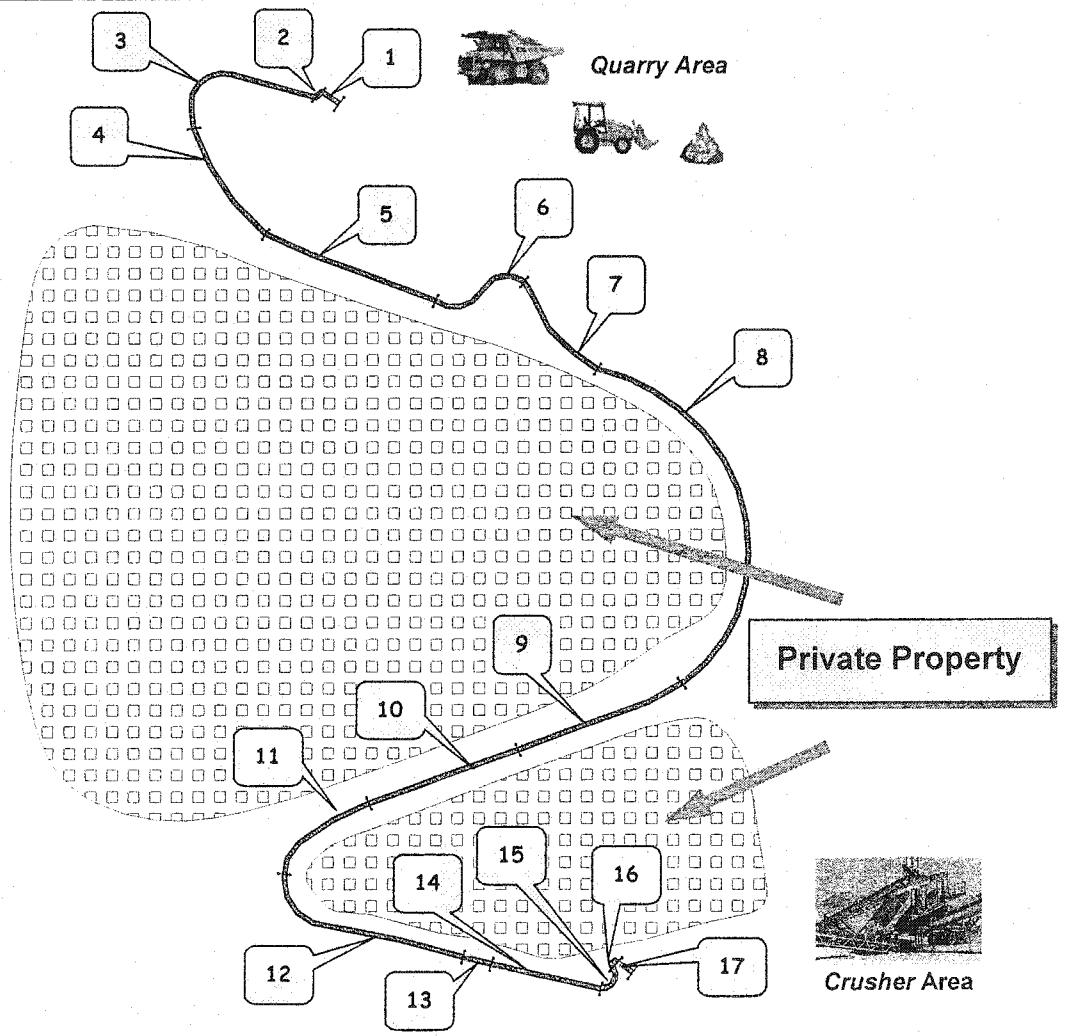


Figure 3.30 Original Haul Road Layout and Characteristics

Alternate Haul Road Segments

Segment	Length (m)	Grade (%)
1	20	0
2	3	0
3	125	6
4	110	0
5	500	-6
6	100	-9
7	185	0
8	25	7
9	99	10
10	26	1
11	3	0
12	18	0
	1214	

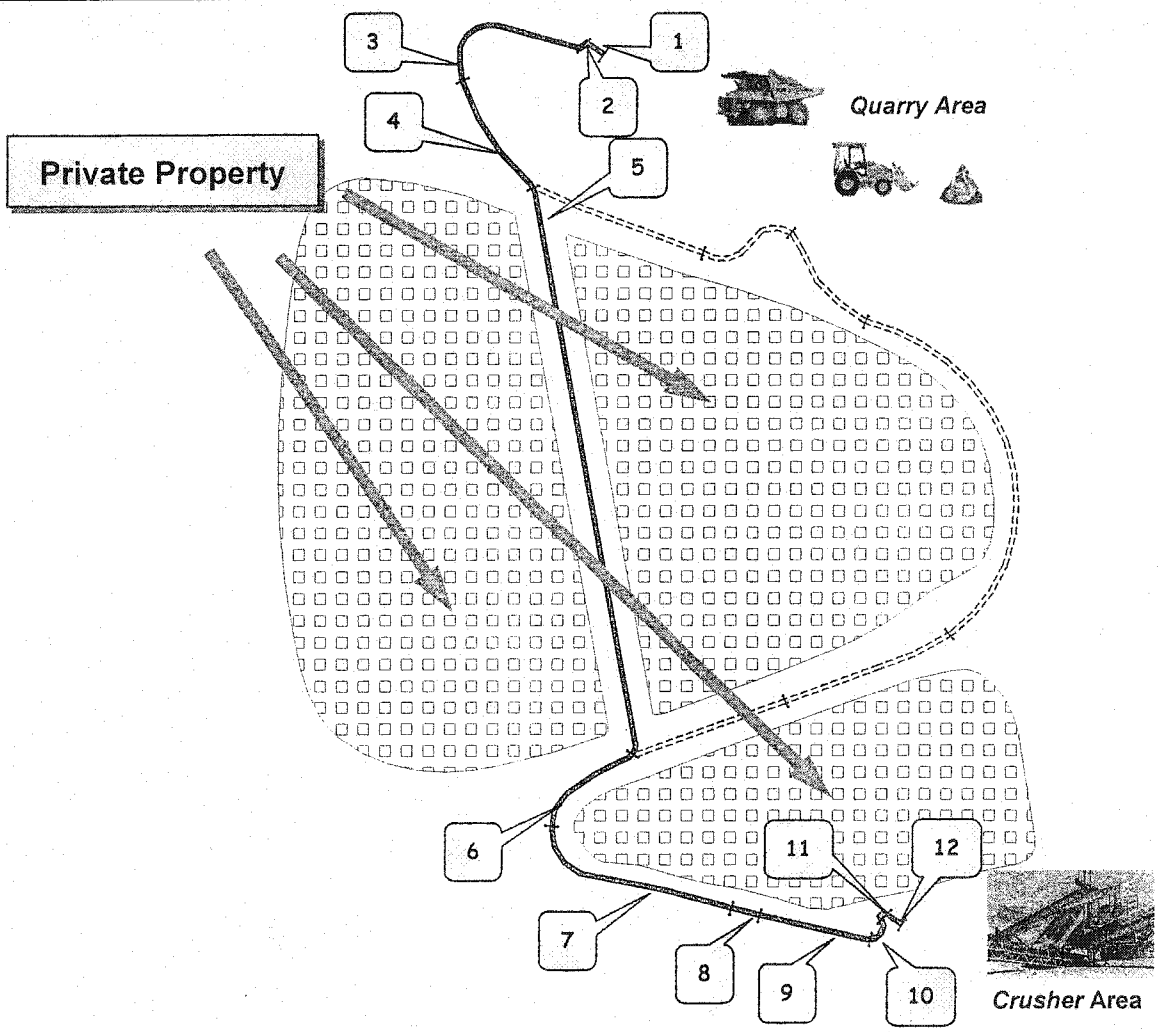


Figure 3.31 Alternate Haul Road Layout and Characteristics

Table 3.7 FPC Input Data

Operator Efficiency :	95 %
Loader Availability (988F II) :	94 %
Hauler Availability (769D) :	96 %
Hauler Availability (771D) :	96 %
Loader Passes (769D) :	3
Loader Passes (771D) :	4
Dump and Maneuver Time :	1.5 min.
Loader Cycle Time :	0.5 min.

Table 3.8 EMSP Input Data

	Activity Type	Theoretical Time (Min.)	Estimated Time (Min.)	Distribution	Parameters		
					Lower Limit (Min.)	Mode (Min.)	Upper Limit (Min.)
Scenario_1	Load	--	--	Uniform	1.5	--	1.7
	Haul	4.4	5.14	Triangle	4.36	4.59	5.05
	Dump	--	--	Uniform	1.4	--	1.6
	Return	3.54	3.87	Uniform	3.65	--	4.22
Scenario_2	Load	--	--	Uniform	2.0	--	2.2
	Haul	4.87	5.88	Triangle	4.57	4.81	5.29
	Dump	--	--	Uniform	1.4	--	1.6
	Return	3.74	3.99	Uniform	3.72	--	4.31
Scenario_3	Load	--	--	Uniform	1.5	--	1.7
	Haul	3.78	4.43	Triangle	2.84	2.99	3.29
	Dump	--	--	Uniform	1.4	--	1.6
	Return	3.74	3.99	Uniform	2.75	--	3.18
Scenario_4	Load	--	--	Uniform	2.0	--	2.2
	Haul	4.32	5.63	Triangle	3.05	3.21	3.53
	Dump	--	--	Uniform	1.4	--	1.6
	Return	3.35	3.47	Uniform	2.83	--	3.28

Upon completion of the data entry, *EMSP* performs simulations for the four scenarios referred to earlier. The required hours to accomplish the job, productivity, cost per ton and total direct cost obtained from *EMSP* and *FPC* are listed in Table 3.9. The analysis results indicate that both *FPC* and *EMSP* determined that Scenario_4 is the optimum fleet scenario that satisfies the user-stated objective of minimum project duration. It is interesting to note that, for the selected scenario, *EMSP* estimated the duration to be 465 hours, while *FPC* estimated it to be 527 hours. In terms of cost, Scenario_4 also provides the minimum total direct cost, estimated to be $168,795 and $182,338 utilizing *EMSP* and *FPC*, respectively. With respect to the financial viability, the analysis performed provides useful information, necessary for assessing such viability. For example, using 771D haulers, the difference in cost between the original and the alternate haul roads is estimated to be $42,827 based on the analysis performed using *FPC*, and $44,286 based on the analysis performed using *EMSP.* This cost forms an envelope for the fee required to use the private property during the project execution (approximately 2.9 months) and the cost of preparing the access road. Further, the financial viability should account for the reduction in the indirect cost due to the reduction in the project duration.

The results obtained from *FPC* and *EMSP* indicate a close agreement (differences range from approximately 11 to 13%). It should be noted that *FPC* lacks the ability to quantify and model the uncertainty that may exist in earthmoving operations without requesting the user to introduce subjective

judgement. This is particularly true when the user of *FPC* is requested to specify the availability percentages of loaders and haulers. The difference in the estimated durations shown in Table 3.9 could be attributed to the subjectivity referred to earlier.

Table 3.9 Analysis Results

	FPC				EMSP				*AE (%)
	Dur.	Prod. (Ton/Hr)	Direct Cost ($)	Cost per Unit ($/Ton)	Dur.	Prod. (Ton/Hr)	Direct Cost ($)	Cost per Unit ($/Ton)	
Scenario_1	813	277	274,759	1.22	732	307	259,860	1.16	11.1
Scenario_2	651	346	225,165	1.00	587	383	213,081	0.95	10.9
Scenario_3	637	353	215,205	0.96	570	395	202,350	0.89	11.7
Scenario_4	527	427	182,338	0.81	465	484	168,795	0.75	13.3

* $AE=[(FPC_{Dur}- EMSP_{Dur})/EMSP_{Dur}]*100$

3.7 Summary

This chapter presented the developed simulation engine (*EMSP*), dedicated for modeling earthmoving operations. *EMSP* represents a main component in the *SimEarth* system. The engine has been designed utilizing discrete event simulation, object-oriented simulation (OOS) and three-phase simulation. *EMSP*'s main and auxiliary classes have been listed, showing their characteristics and designated tasks.

A numerical example of an actual case has been analyzed using the Caterpillar's software *FPC* and the developed *EMSP*. To enable a comparison, the analysis

performed using *EMSP* was limited to the features available in *FPC*. The results indicate a close agreement between the two outputs obtained from *FPC* and *EMSP*. In addition, *EMSP* provides a vehicle for modeling uncertainties that may exist in earthmoving operations in a reliable manner without introducing subjective judgement.

Chapter 4

DATABASE AND COST APPLICATIONS

4.1 Introduction

This chapter focuses on two components of the proposed *SimEarth* system: 1) earthmoving equipment database (*EDA*) and 2) equipment cost application (*ECA*). Earthmoving equipment database is dedicated to store all earthmoving equipment related information, including their own characteristics and possible attachments in order to be retrieved through an easy to use interactive user friendly environment. Section 2 of this chapter provides a review of the types of database models. It also describes the various design and implementation stages of *EDA*.

Equipment cost application is designed for estimating earthmoving hourly owning and operating costs considering their different categories. Section 3 of this chapter presents an overview of the main categories of equipment cost along with their components. It illustrates required calculations, which are carried out by *ECA* to obtain these cost components. Finally, a numerical example is presented to demonstrate the use of the developed application and its essential features.

4.2 Equipment Database Application

Equipment database application (*EDA*) is designed to store all equipment

characteristics which can be utilized in earthmoving operations along with their attachments. It also stores haulers' allowable speed in order to estimate haulers' travel time as will be described later in Chapter 5. The user can obtain any information related to his/her equipment fleet scenario(s) in a comprehensive paper format (see Appendix B). The information includes: 1) equipment characteristics (model, tires, flywheel, top speed, number of cylinders, rated load, etc.); 2) equipment attachments (bucket type, bucket capacity, bucket width, etc.); and 3) equipment range dimensions (maximum height, ground reach, digging depth, etc.). Equipment information was extracted from the Caterpillar performance handbook (1997). It also allows for the inclusion of equipment manufactured by other equipment suppliers beyond CAT.

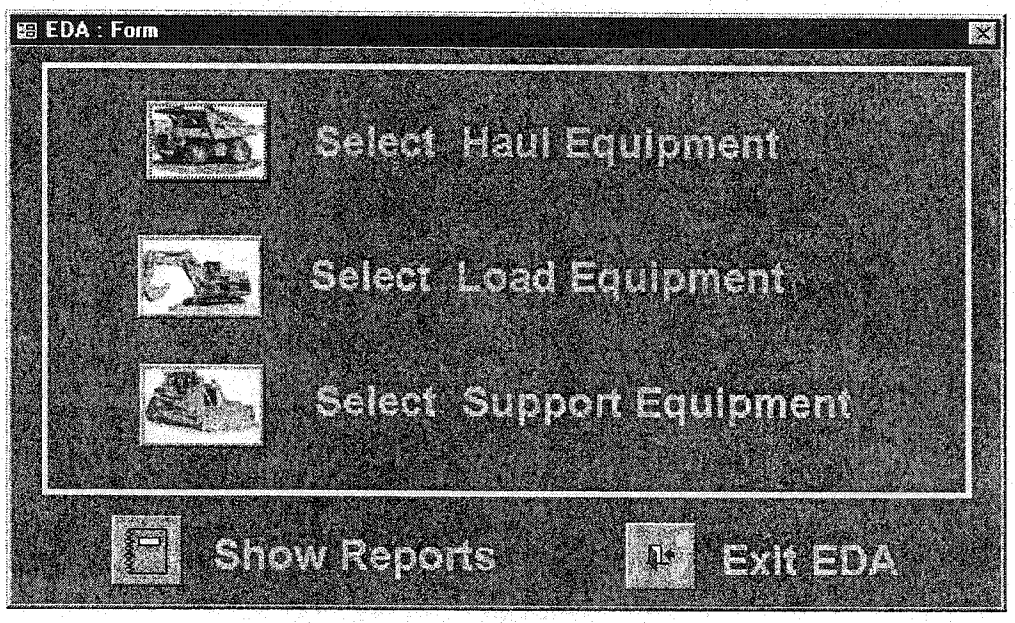


Figure 4.1 EDA Main Menu

The *EDA* is essentially a relational database that has been implemented using MS Access 97. *EDA*'s equipment is classified into three major categories,

according to the process the equipment performs. These categories are: 1) haul equipment; 2) load equipment; and 3) support equipment (see Figure 4.1).

Earthmoving equipment may perform another process (as a minor use) in addition to its main use. For example, a track loader can be used as haul equipment (minor use) to haul earth for short distances in addition to its main use as load equipment. Accordingly, equipment can be shared among the *EDA*'s major categories as shown in Figure 4.2. The following subsections provides an overview of the different database models.

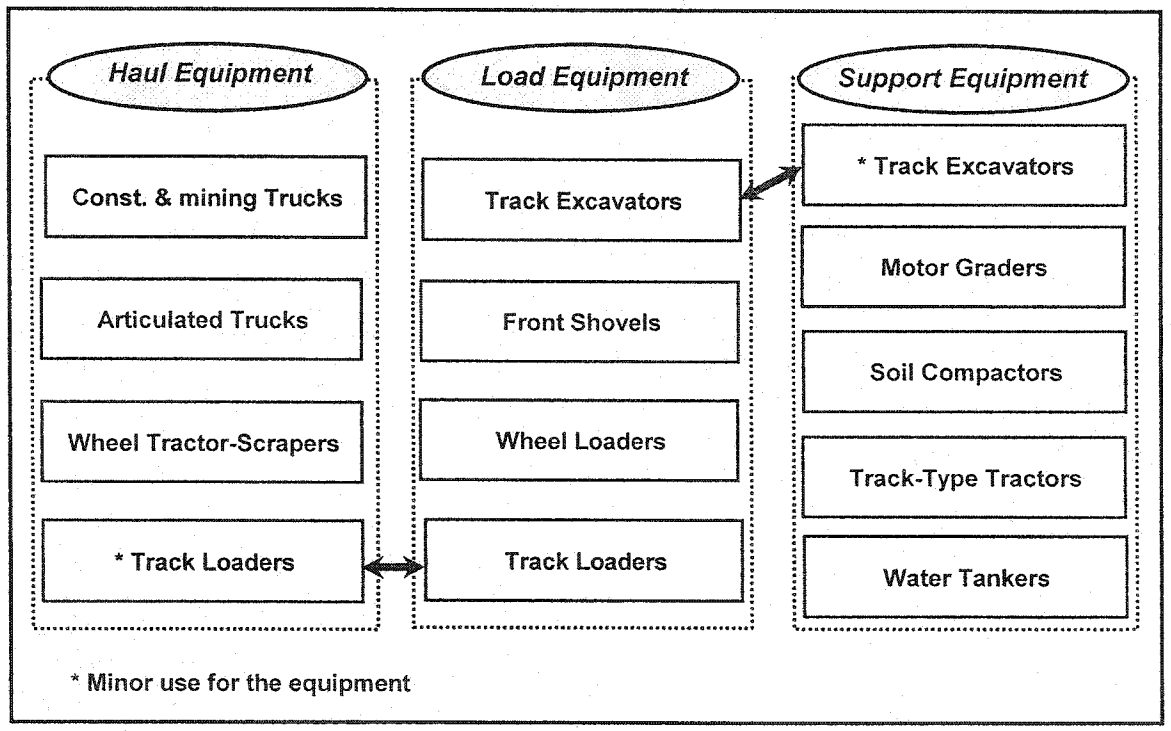


Figure 4.2 EDA Major Categories

4.2.1 Database Models

Computer databases are usually developed in order to reduce the human effort

in extracting information from paper-based sources. Database management systems (DBMS) are used to assist in data storage and retrieval in an interactive manner. Databases are classified into different models according to their data structure and processing mechanism. These models are (Johnson 1997, Elmasri and Navathe 1994): 1) relational database; 2) object-oriented database; 3) deductive database; 4) hierarchy database; and 5) network database models.

In a "*Relational database Model*", data are organized in tables. Each table represents an entity of the application under consideration (Johnson 1997), whereas, tables' columns and rows represent entities' attributes and instants, respectively. Relationships link entities (tables) by including one of the entity attributes in the other entities. For example, the attribute `ExcavatorModel/Source` is made common in the two entities of `Front Shovels` and `Buckets` as shown in Figure 4.3.

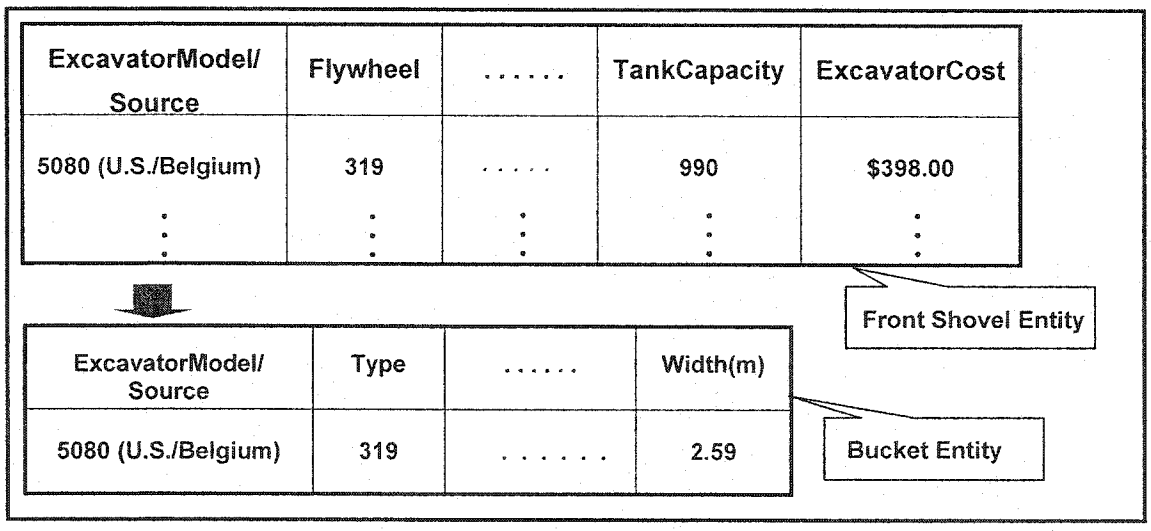


Figure 4.3 Relationship in a Relational Database Model

In an "*Object-Oriented database Model*", the application entities are represented via classes. Entity properties are captured by class attributes and handled by class methods. All object-orientation features, described earlier in Chapter 3 including inheritance, dynamic data structure and polymorphism, can be employed in object-oriented databases. For example, a `Load_Equip` class can be utilized as a superclass for `Front_Shovels`, `Track_Excavators`, `Wheel_Loader` and `Track_Loader` classes. All attributes and methods of `Load_Equip` class are inherited by its subclasses. Subclasses may, in addition, have their own attributes and methods. Relationships are defined employing the logical containment concept. For example, a bucket entity is defined as one of the `Load_Equip` class attributes, implicitly defining the relationship (see Figure 4.4).

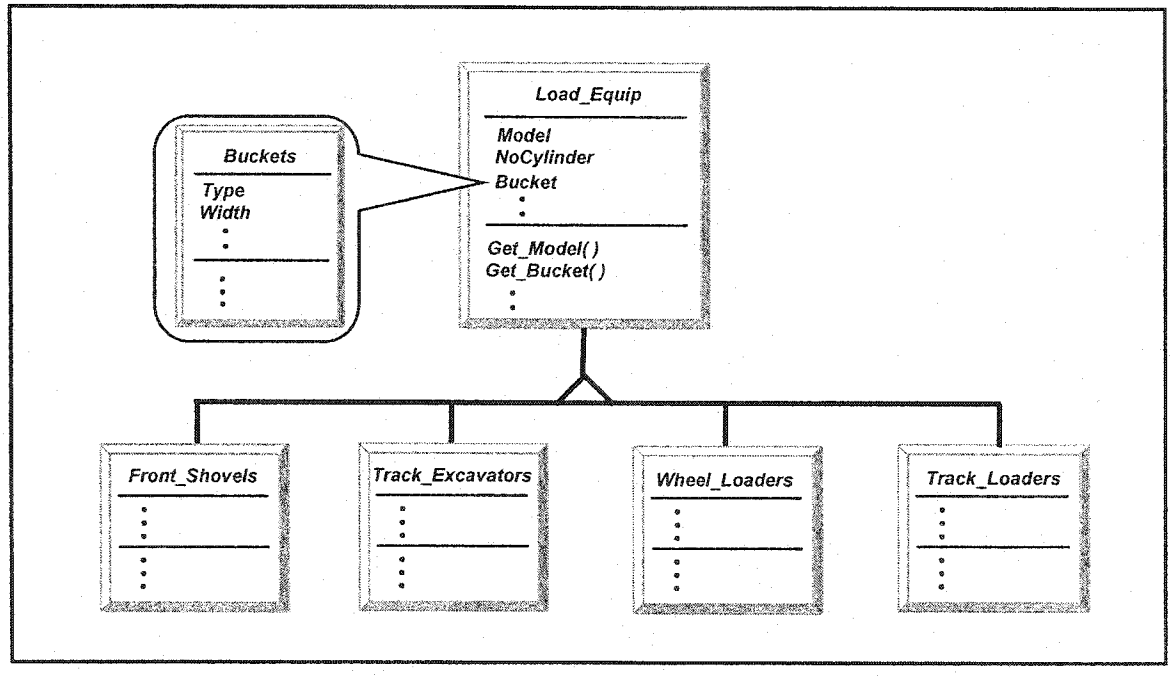


Figure 4.4 Inheritance in an Object-Oriented Database Model

In a "*Deductive database Model*", data and relationships are defined into axioms. It has an ability to generate new data combinations by declaring different rules on the stored data (Johnson 1997). For example, consider two instants of `Track_Tractors` entity (D3C Series III and D4C Series III) that are configured by power angle and tilt (PAT) blade of `Blade` entity. The following two axioms are defined `TractorHasBlade` (D3C Series III, PAT) and `TractorHasBlade` (D4C Series III, PAT). The axiom `SharedBladeTractors`(D3C Series III, D4C Series III) is deducted automatically from the existing facts (see Figure 4.5).

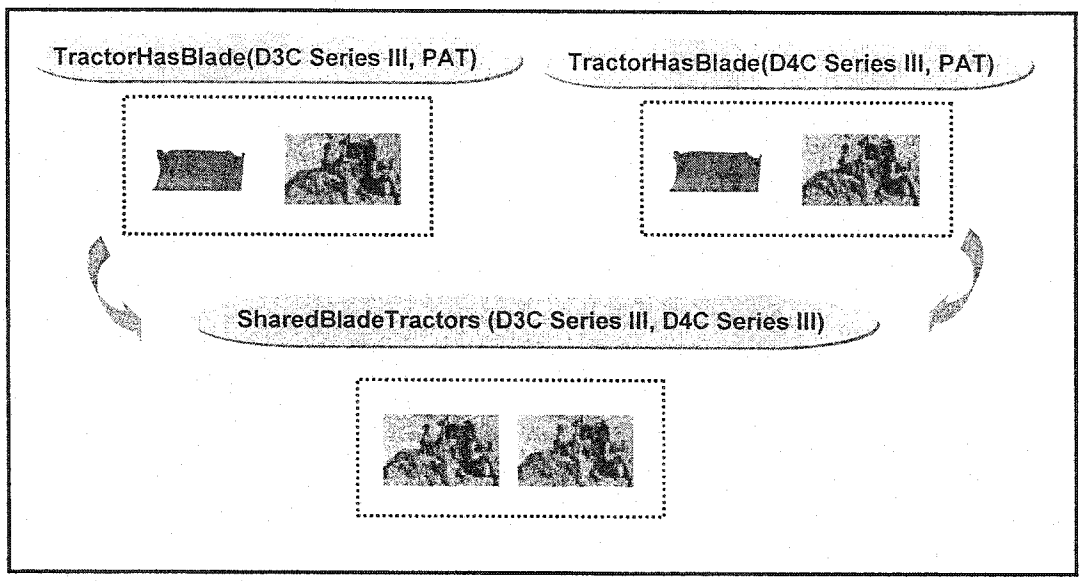


Figure 4.5 Data Generation in a Deductive Database Model

In a *"Hierarchy database Model"*, a tree structure of the application is constructed to capture the relationship among its entities (Johnson 1997). For example, haul equipment category is divided into four divisions which represents their equipment (construction and mining trucks, articulated trucks, wheel tractor-scrapers and truck loaders). Each of these divisions is further divided into

subdivisions which represent their model as shown in Figure 4.6. Data are organized in rows that provide the relation logic among the application entities.

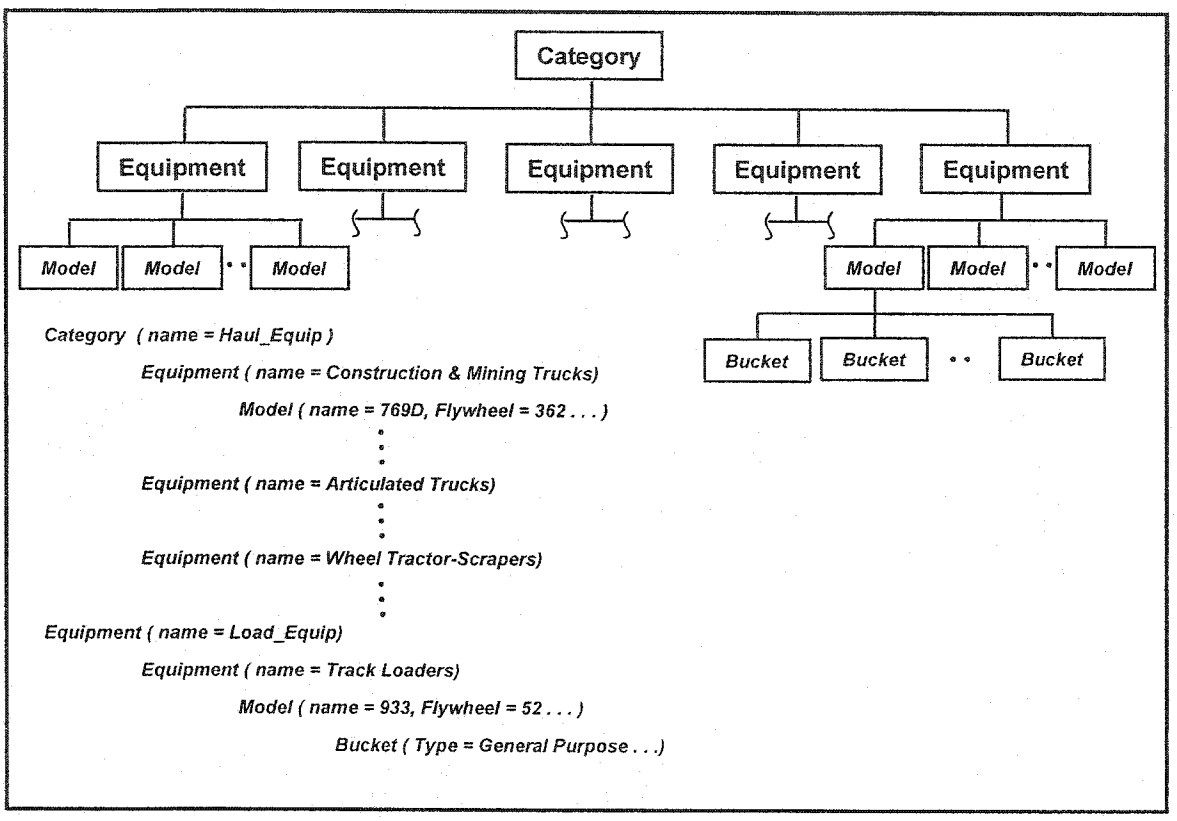


Figure 4.6 Tree Structure in a Hierarchy database Model

In a "*Network database Model*", the hierarchy (in the hierarchical database model) is replaced by a graph to overcome the difficulty in representing shared entities among different higher levels in the hierarchy (Johnson 1997). For example, `Track Loaders` entity is shared between haul equipment and load equipment. This shared entity is maintained by applying intersecting chain as illustrated in Figure 4.7.

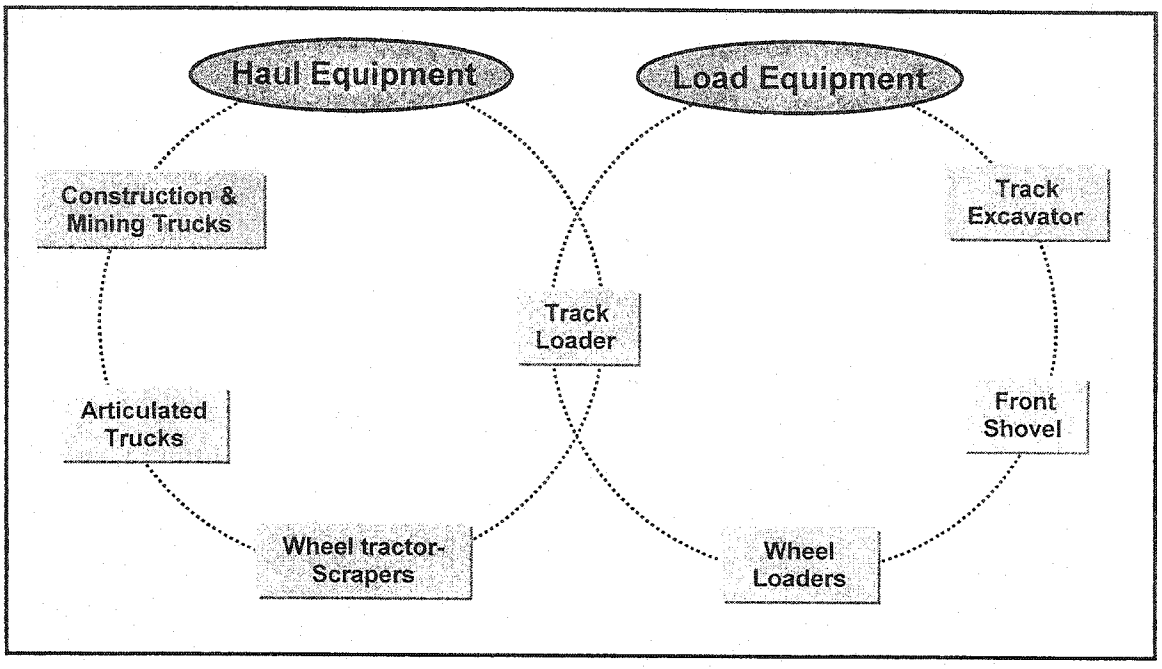


Figure 4.7 Entity Sharing in a Network Database Model

EDA is modeled utilizing relational database to provide: 1) simple format (collection of tables); 2) effective user interaction; and 3) integration with the rest of the components of *SimEarth* in MS environment.

4.2.2 Development of EDA

EDA has been developed to ease the data retrieval process in *SimEarth*. It consists essentially of two relational databases; one for the characteristics of equipment and the second for the haulers' allowable speeds. The latter database feeds *HTTA* with haulers' allowable speeds needed for estimating haulers' travel time. The equipment database feeds *SimEarth* with the characteristics of the

equipment being considered, including possible attachments. The design process of the equipment database has been carried out in four stages as proposed by Elmasri and Navathe (1994) as shown in Figure 4.8. These stages are: 1) requirements collection and analysis; 2) conceptual design; 3) logical design; and 4) physical design. In the requirements collection and analysis stage, all earthmoving equipment data including characteristics, attachments and range dimensions were extracted from the Caterpillar Performance Handbook (1997). In addition, the functional requirements were set to: 1) provide the user with equipment data in an easy to use manner; 2) provide a link with *ECA* in order to estimate the hourly owning and operating costs for the selected equipment and update it in *EDA*; and 3) generate a number of output reports in a paper format.

In the conceptual design stage, schema was created by classifying equipment into three major categories: haul equipment; load equipment; and support equipment. Each category was further broken down to a set of sub-databases as shown in Figure 4.2. Accordingly, 11 sub-databases were generated and developed within those categories. It should be noted that certain equipment have dual functions (e.g. track loaders and track excavators). For example, the major use of track loader is a load equipment and its minor use is a haul equipment. The development is carried out by constructing their respective entity relation (ER) diagrams, specifying attributes and relations among individual entities. MS Access 97 has then been used in the implementation of these sub-databases.

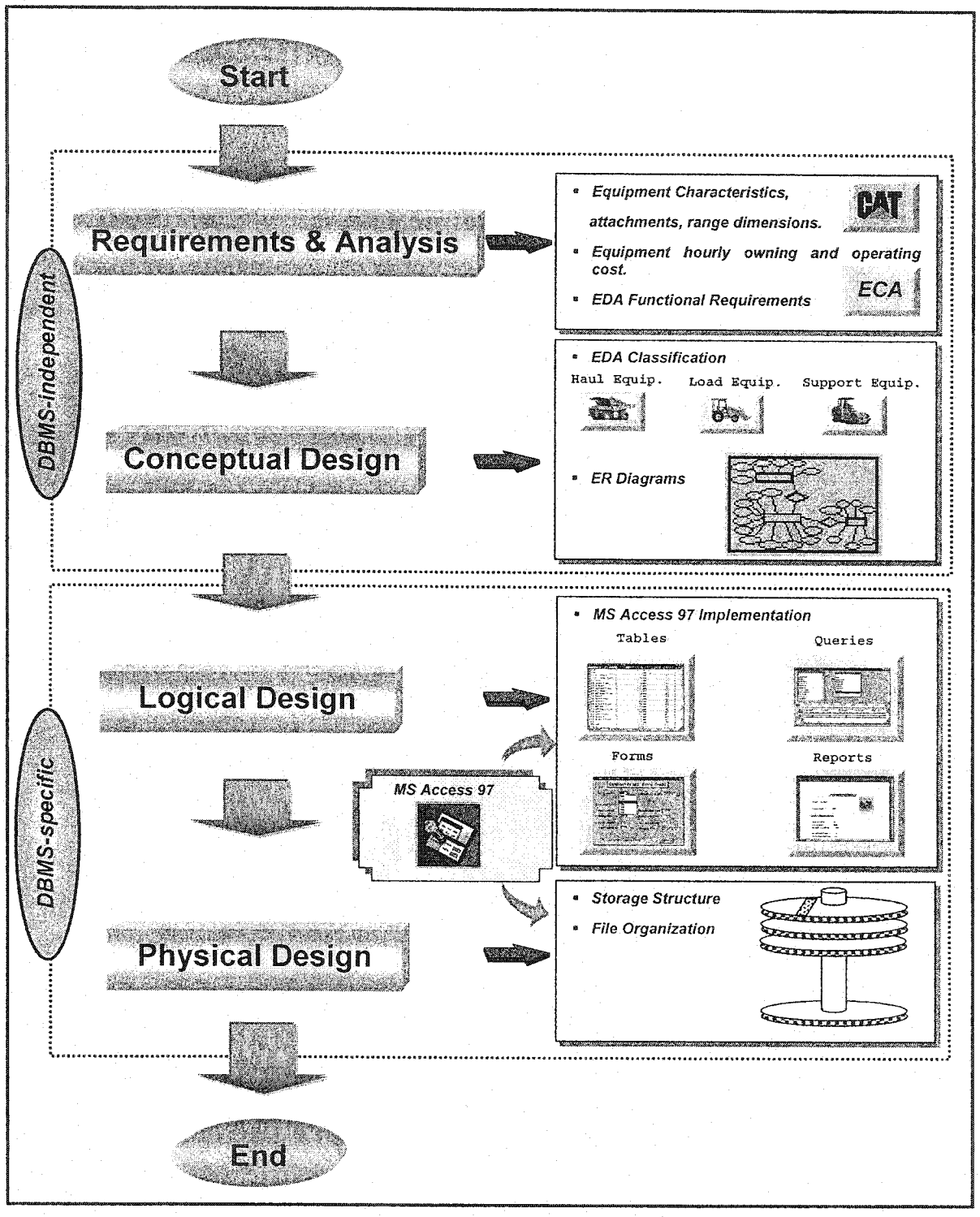


Figure 4.8 EDA Design Process

The developed *EDA*: 1) provides a simple and easy to use environment, 2) facilitates user interaction through a set of specially designed screens and dialog boxes, and 3) supports the integration of the individual components of *SimEarth*. The implementation was carried out by: 1) constructing tables to represent the entities of *EDA*; 2) establishing relationships among the individual entities, 3) developing queries necessary for data processing; 4) designing the user interface; and 5) designing a number of equipment related output reports. In the physical design stage, storage structures and file organization for all involved databases are generated. Since *EDA* has been implemented utilizing MS Access 97 as database management system (DBMS) environment, the physical design is performed automatically in that environment for the eleven equipment sub-databases.

4.2.3 EDA Sub-Databases

EDA is grouped, as stated before, into three major categories (Haul Equipment, Load Equipment and Support Equipment) which capture equipment use. Figures 4.9-a, b and c show the designated main user interfaces for those categories. During the conceptual design stage, *EDA* was divided into eleven sub-databases which represent all earthmoving equipment. ER diagrams for all sub-databases were also constructed to specify entities' attributes and relations. The ER diagram notations of Elmasri and Navathe (1994) were followed as per Figure 4.10.

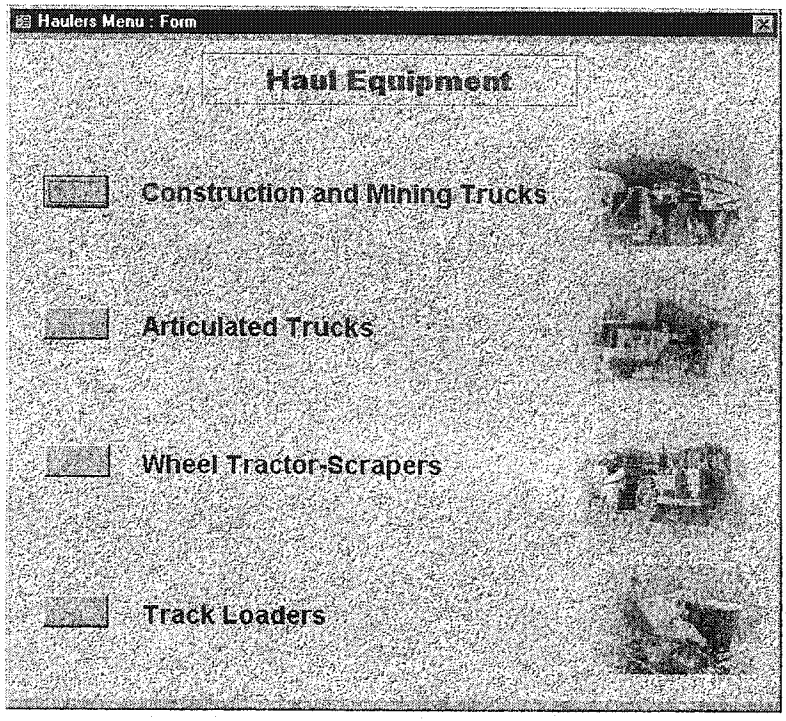


Figure 4.9-a User interface of Haul Equipment Category

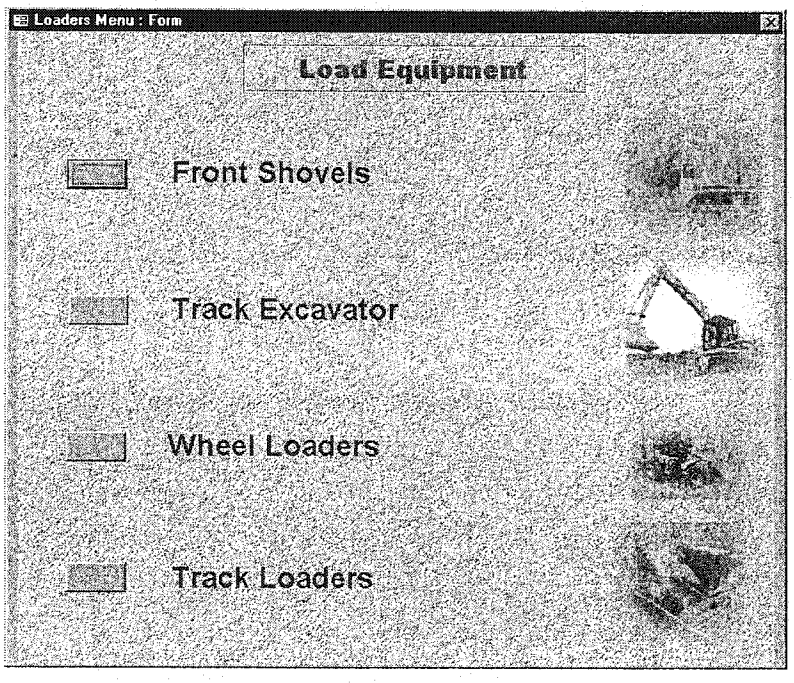


Figure 4.9-b User interface of Load Equipment Category

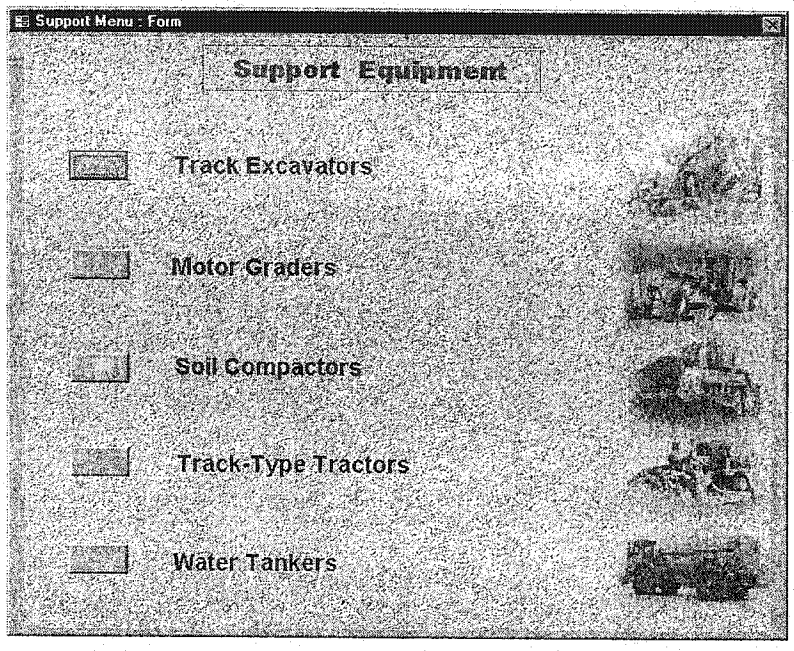


Figure 4.9-c User interface of Support Equipment Category

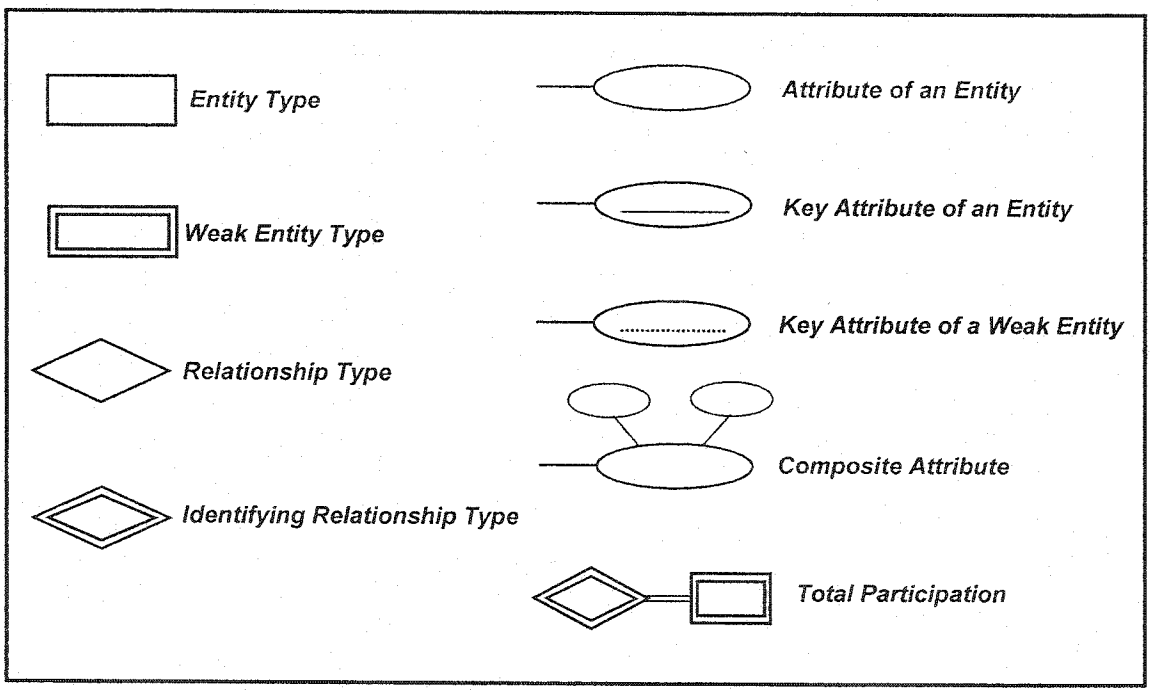


Figure 4.10 ER Diagram Notations

The development process, of the sub-databases, was preformed by: 1) specifying the entities of each sub-database, 2) specifying the attributes of each entity, 3) constructing the relation among each sub-database's entities and 4) implementing user interfaces and customized reports for all designated sub-databases. Appendix B includes some of the reports generated by *EDA*.

The "*Construction and Mining Trucks*" sub-database stores the first equipment data belonging to the haul equipment category. It is a single relation (one table) sub-database (see Figure 4.11). The relation has ten attributes which define all characteristics of construction and mining trucks including truck model, flywheel, rated load, etc. The key attribute of the relation is represented by the `TruckModel` attribute. Figure 4.12 depicts the user interface of the construction and mining trucks sub-database.

Similarly, entity relation diagrams and user interfaces were developed for "*Articulated Trucks*", "*Wheel Tractor-Scrapers*", "*Soil Compactors*", and "*Water Tankers*" (see Figures 4.13 to 4.20).

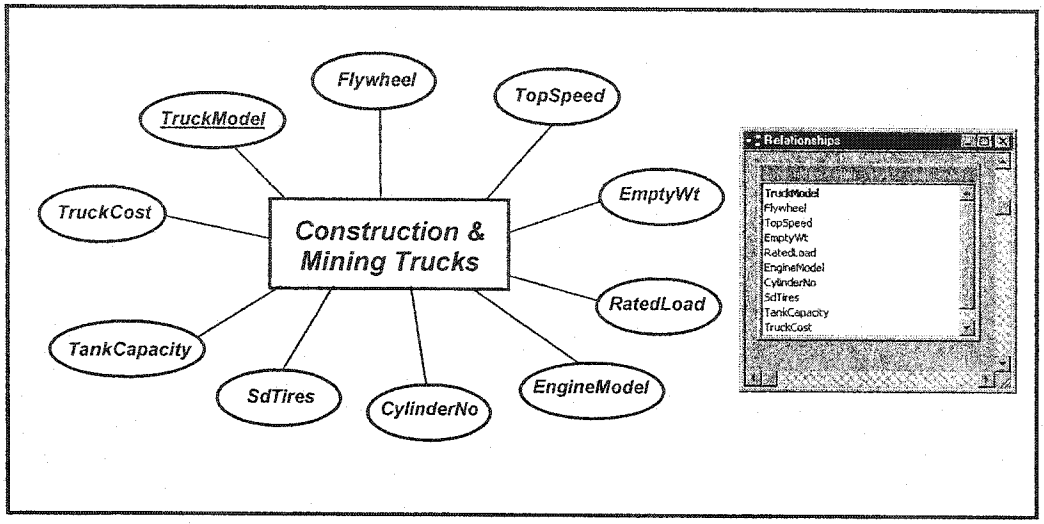


Figure 4.11 ER Diagram for Construction and Mining Trucks Sub-database

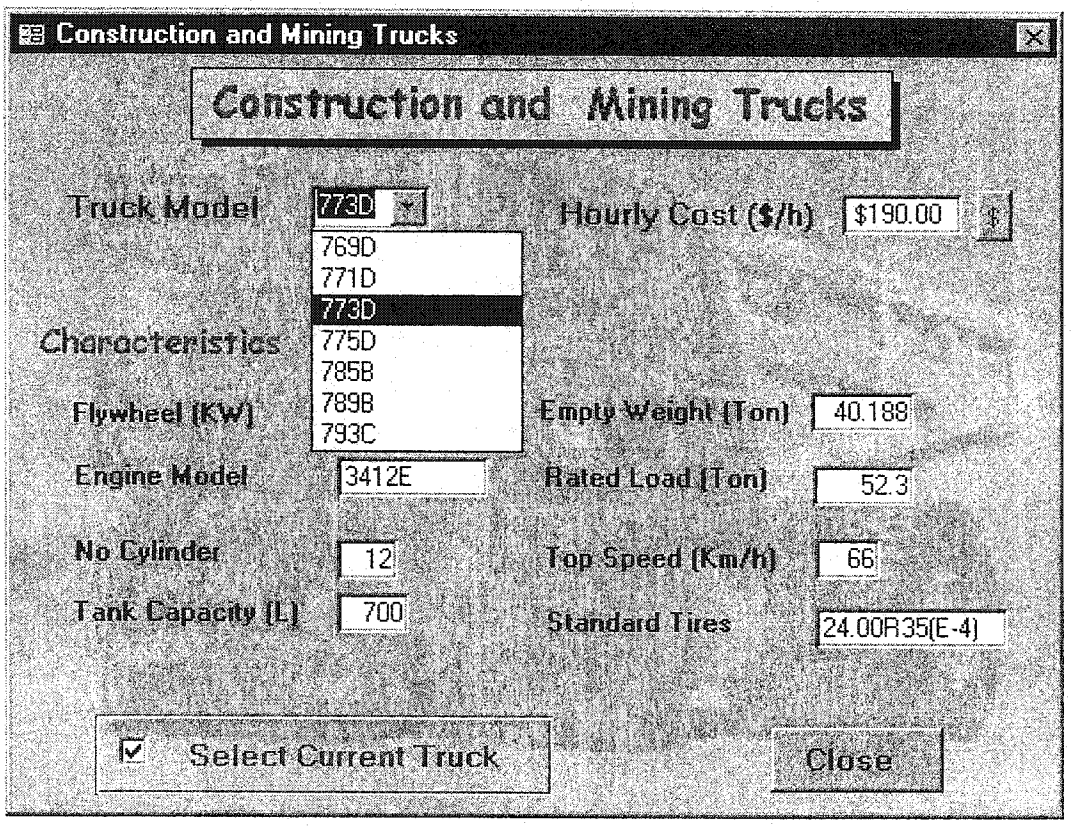


Figure 4.12 Construction and Mining Trucks User Interface

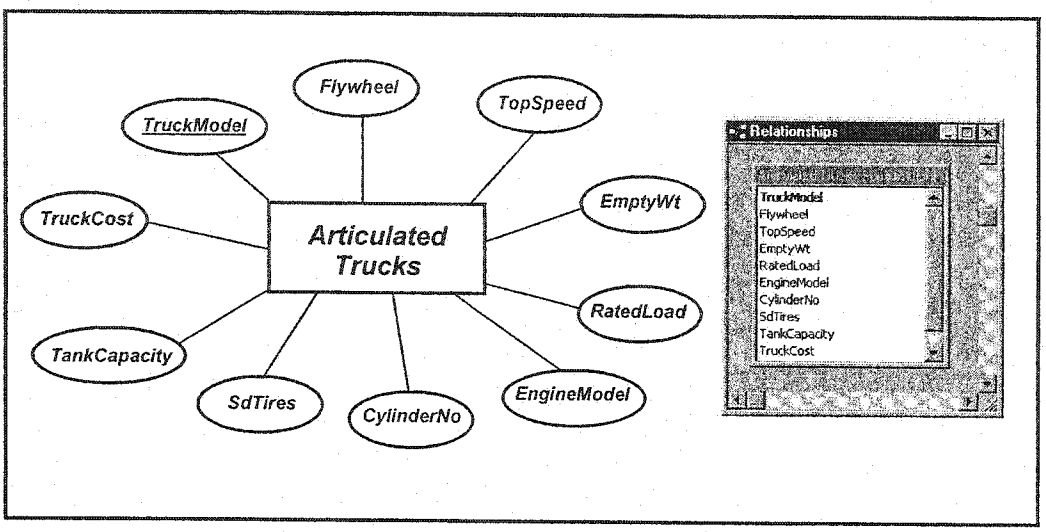


Figure 4.13 ER Diagram for Articulated Trucks Sub-database

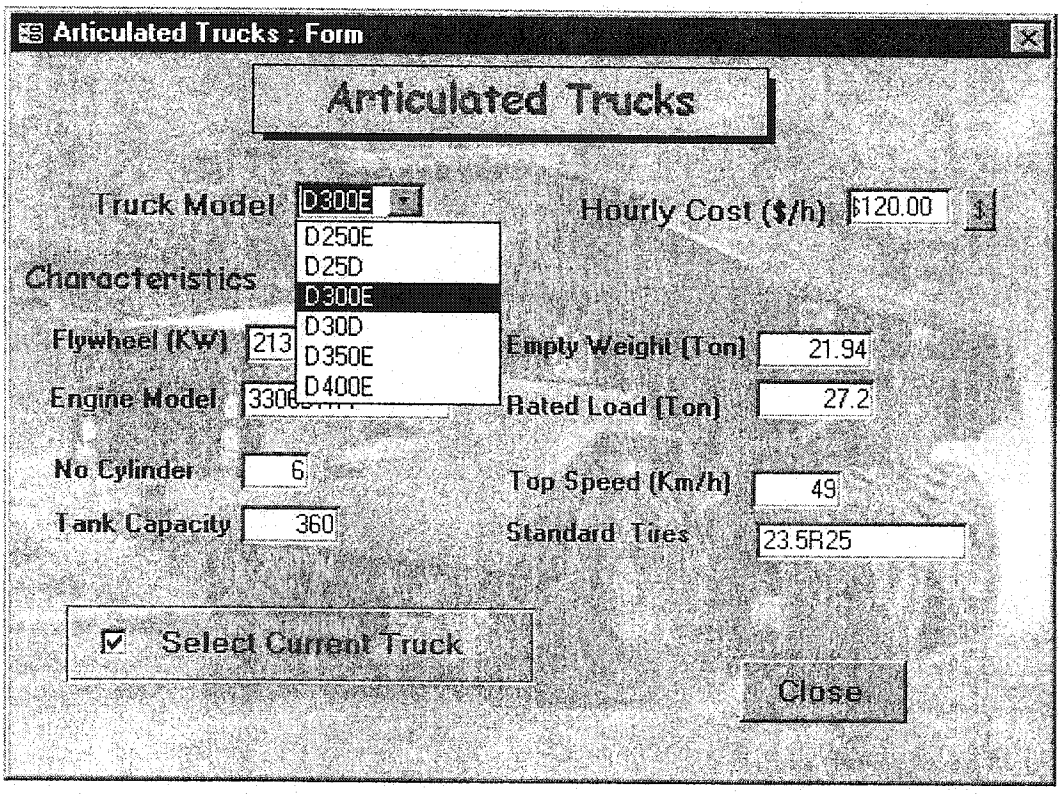


Figure 4.14 Articulated Trucks User Interface

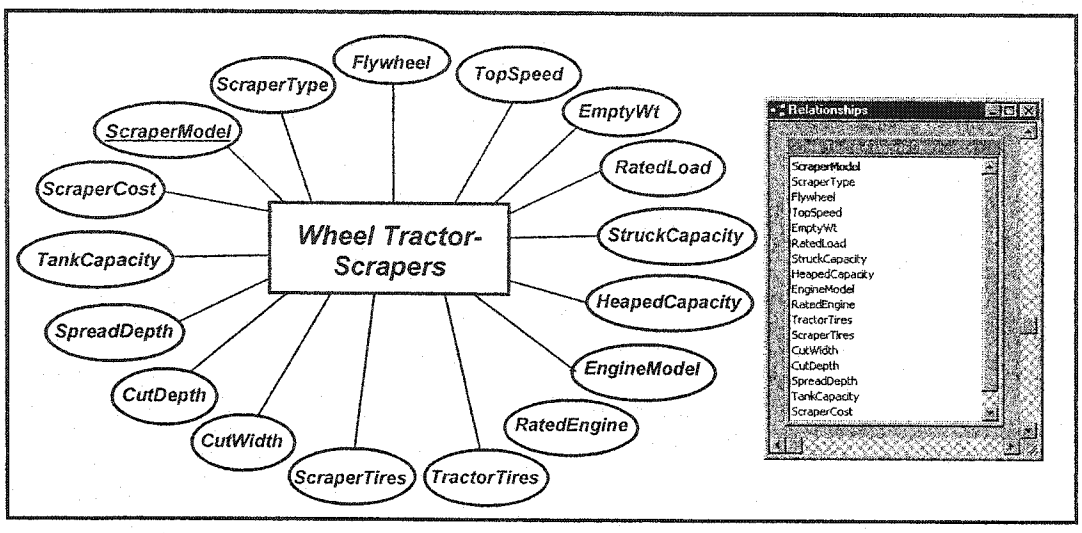


Figure 4.15 ER Diagram for Wheel Tractor-Scraper Sub-database

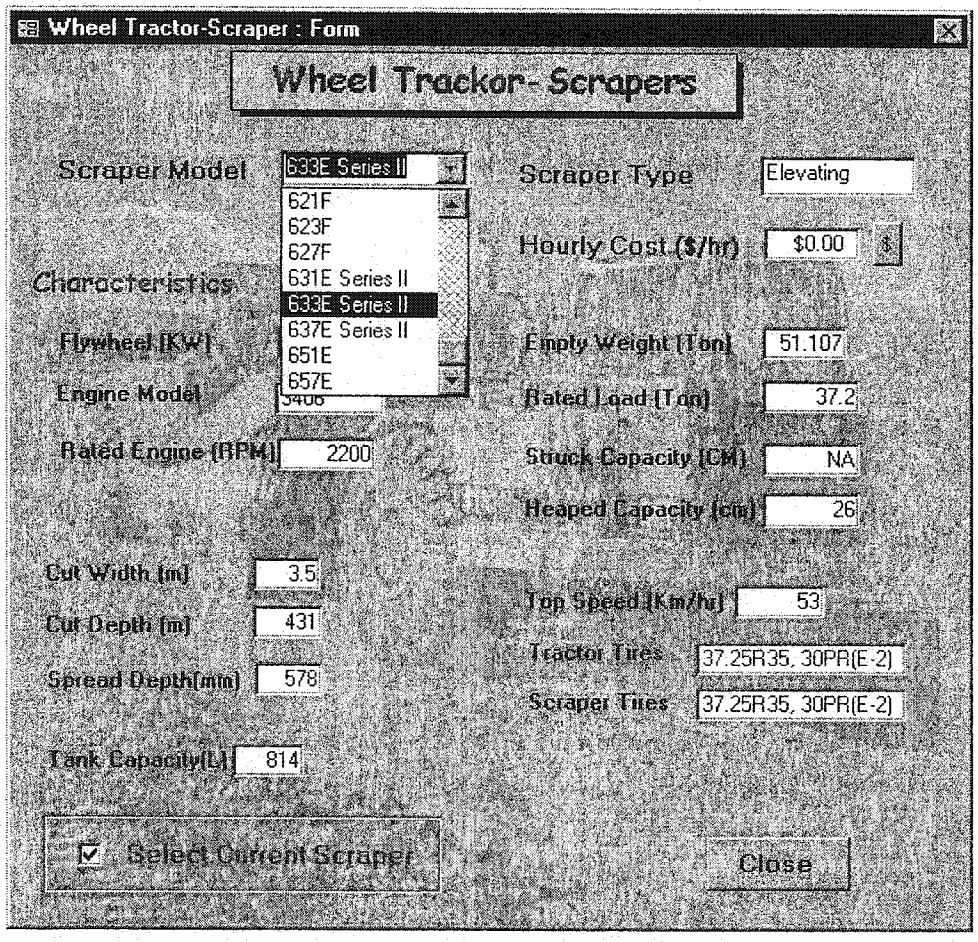


Figure 4.16 Wheel Tractor-Scrapers User Interface

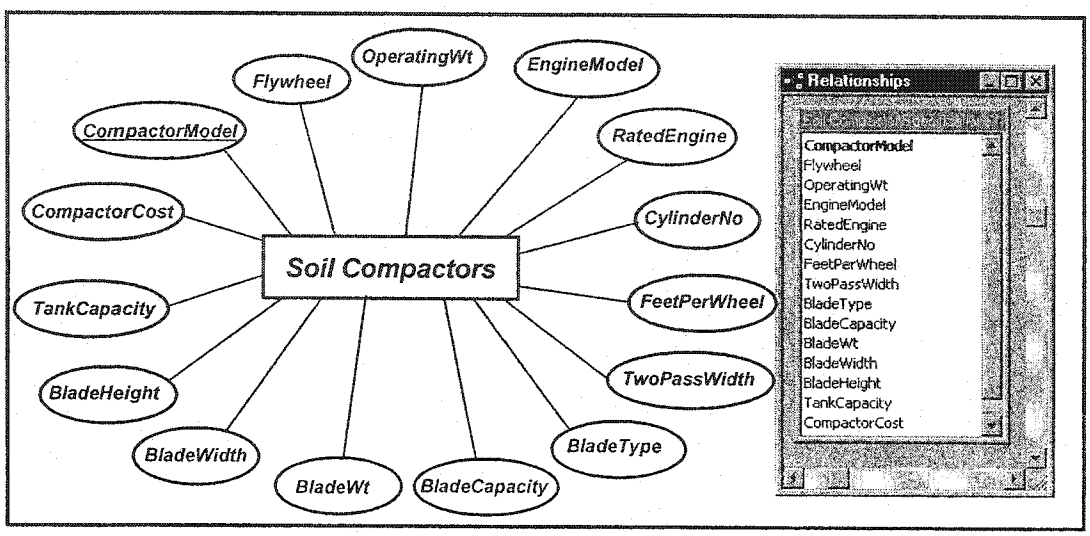


Figure 4.17 ER Diagram for Soil Compactors Sub-database

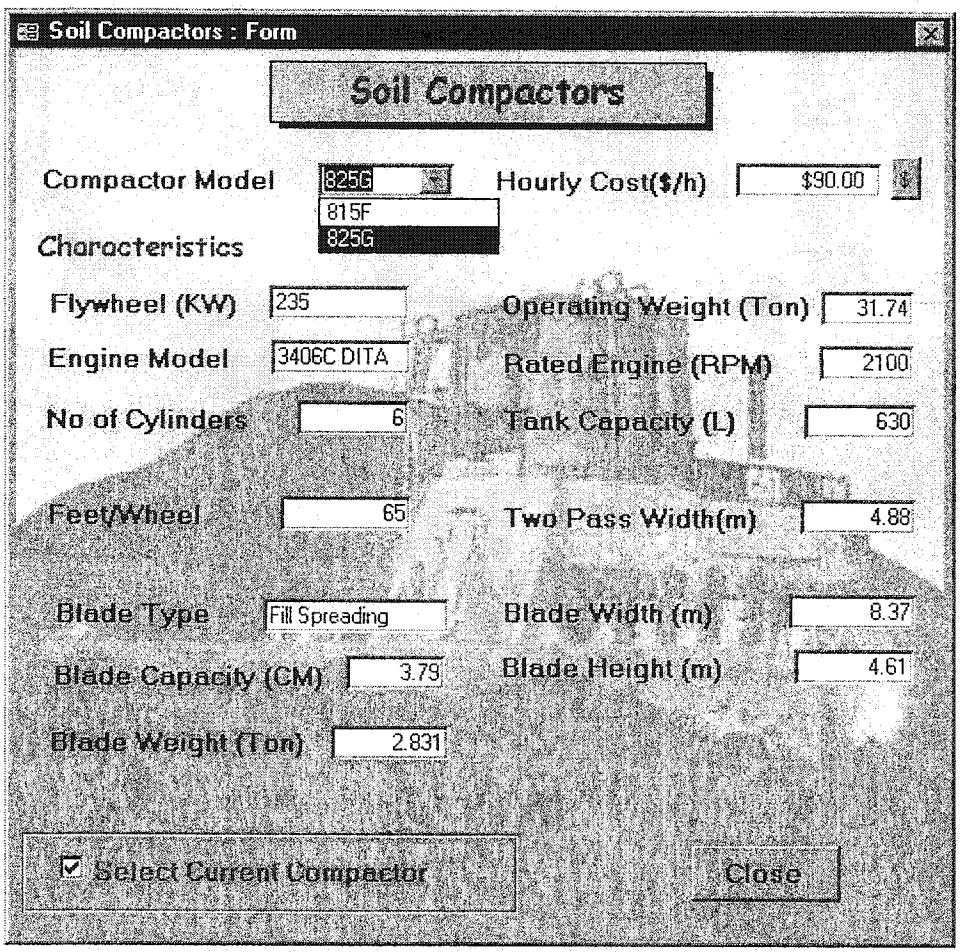


Figure 4.18 Soil Compactors User Interface

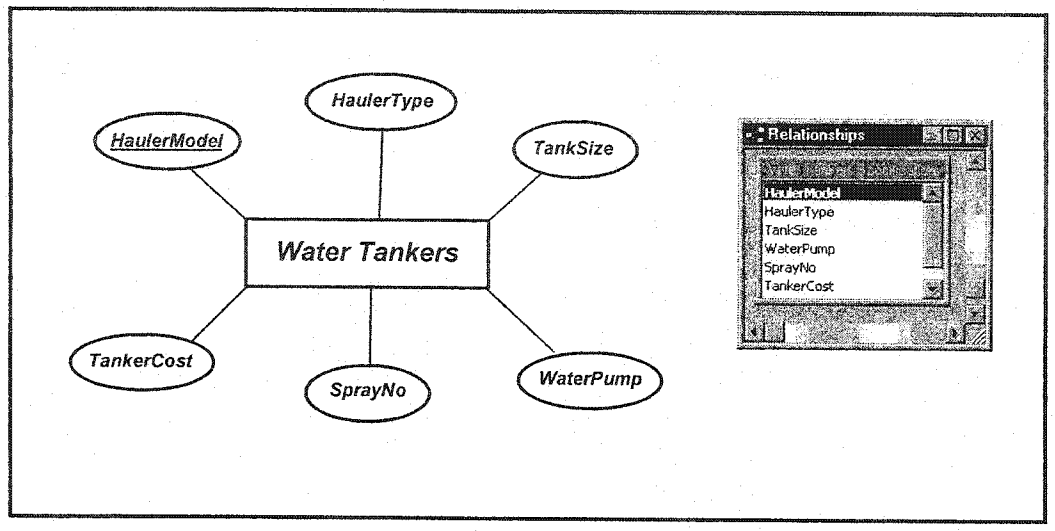


Figure 4.19 ER Diagram for Water Tankers Sub-database

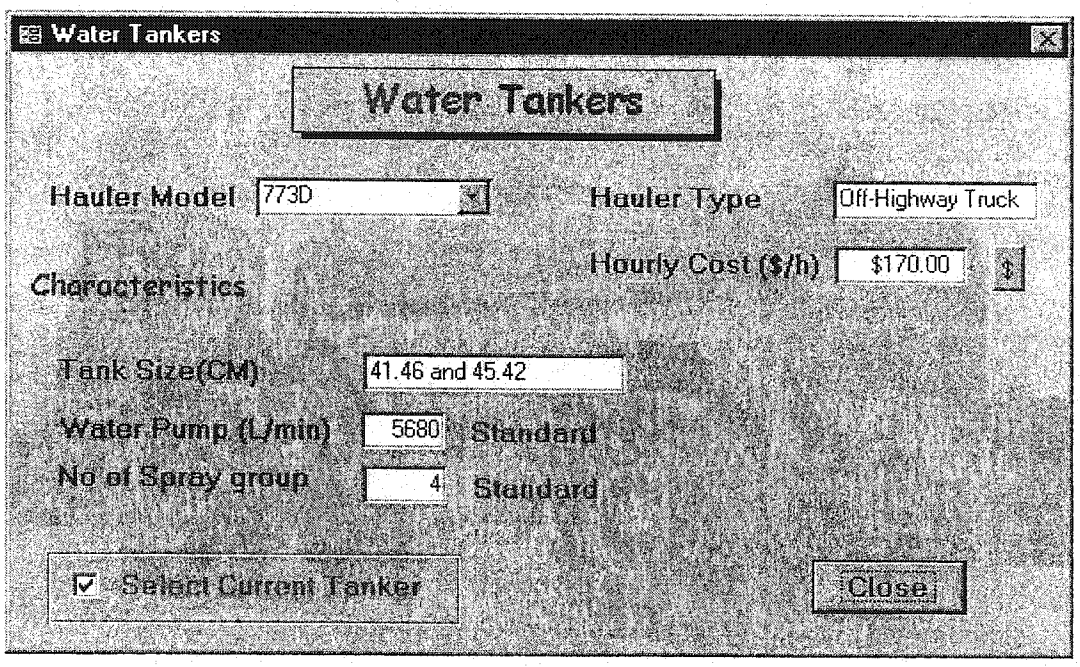


Figure 4.20 Water Tankers User Interface

The "*Front Shovels*" sub-database stores the second equipment data belonging to the load equipment category. It has three entities: `Front Shovels`, `Buckets` and `Range Dimensions` (see Figure 4.21). The entities of `Buckets` and `Range Dimensions` are weak entities, linked to the `Front Shovels` entity via a set of relationships. These entities have total participation constraints through `HAS` (one to many) and `BELONGS` (one to one) relationships, respectively. The `Buckets` entity stores the bucket characteristics (e.g. bucket type, bucket capacity, lift force, etc.). The combination of `ExcavatorModel/Source` and `Type` attributes represents the key attribute for the `Buckets` entity. On the other hand, the `Range Dimensions` entity consists of four attributes with `ExcavatorModel/Source` as a key attribute. It stores the front shovels' range dimensions including maximum height (L), ground reach (H) and digging depth (D). Finally, the `Front Shovels` entity, which consists of nine attributes, stores the characteristics of front shovels including the excavator's model and source, flywheel, number of cylinders, etc. The `ExcavatorModel/Source` represents the key attribute of `Front Shovels` entity. Figure 4.22 depicts the user interface of the front shovel sub-database.

Similarly, entity relation diagrams and user interfaces were developed for "*Track Loaders*", "*Track Excavators*", "*Wheel Loaders*", "*Motor Graders*" and "*Track-Type Tractors*" (see Figures 4.23 to 4.32).

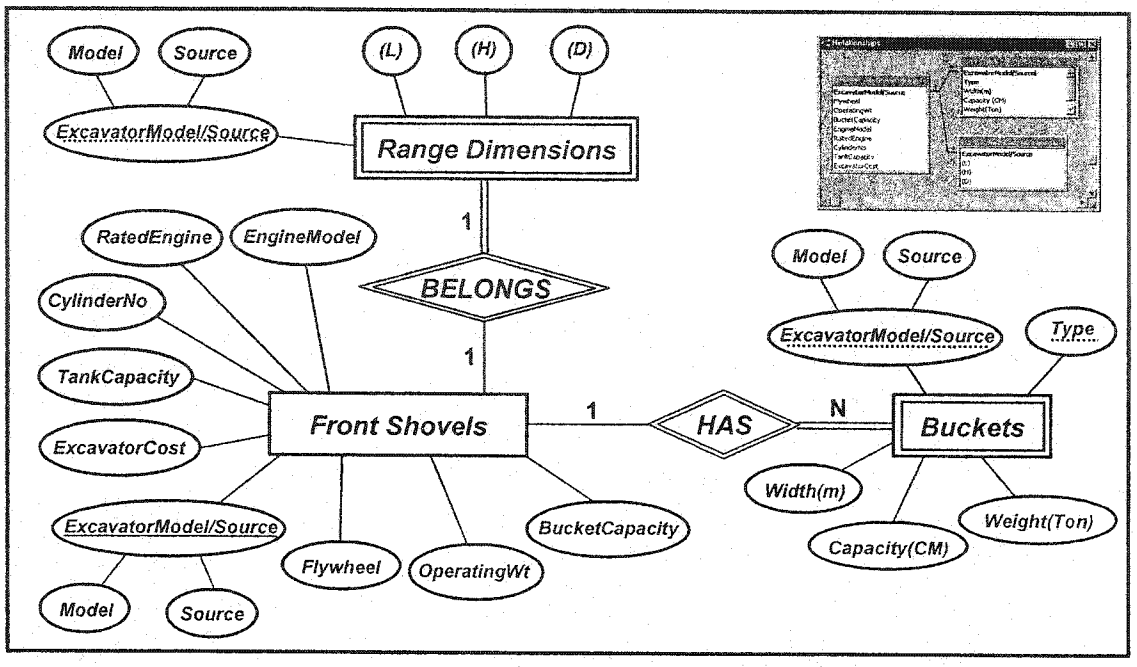


Figure 4.21 ER Diagram for Front Shovels Sub-database

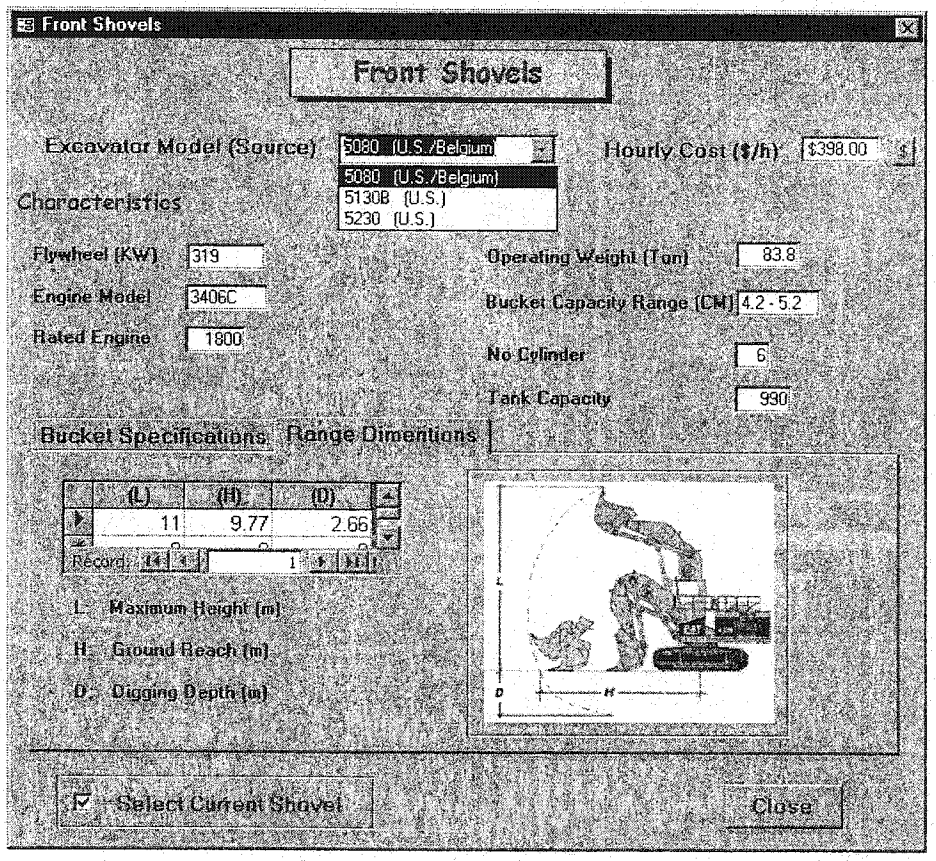


Figure 4.22 Front Shovels User Interface

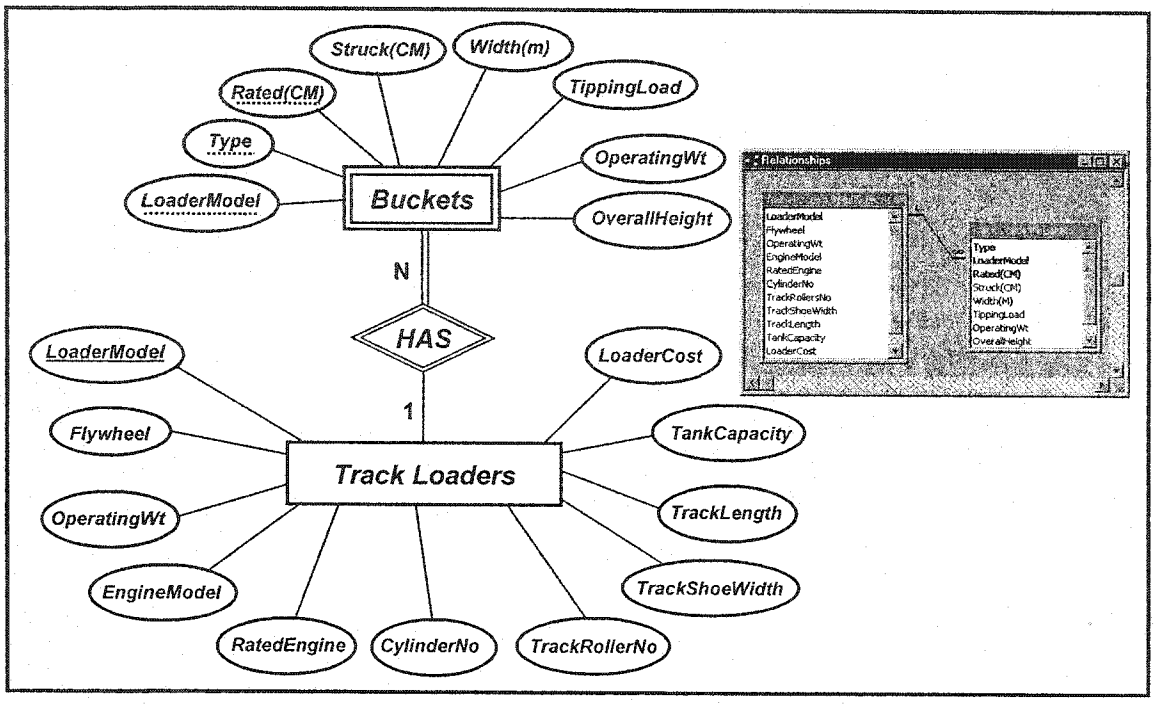


Figure 4.23 ER Diagram for Track Loaders Sub-database

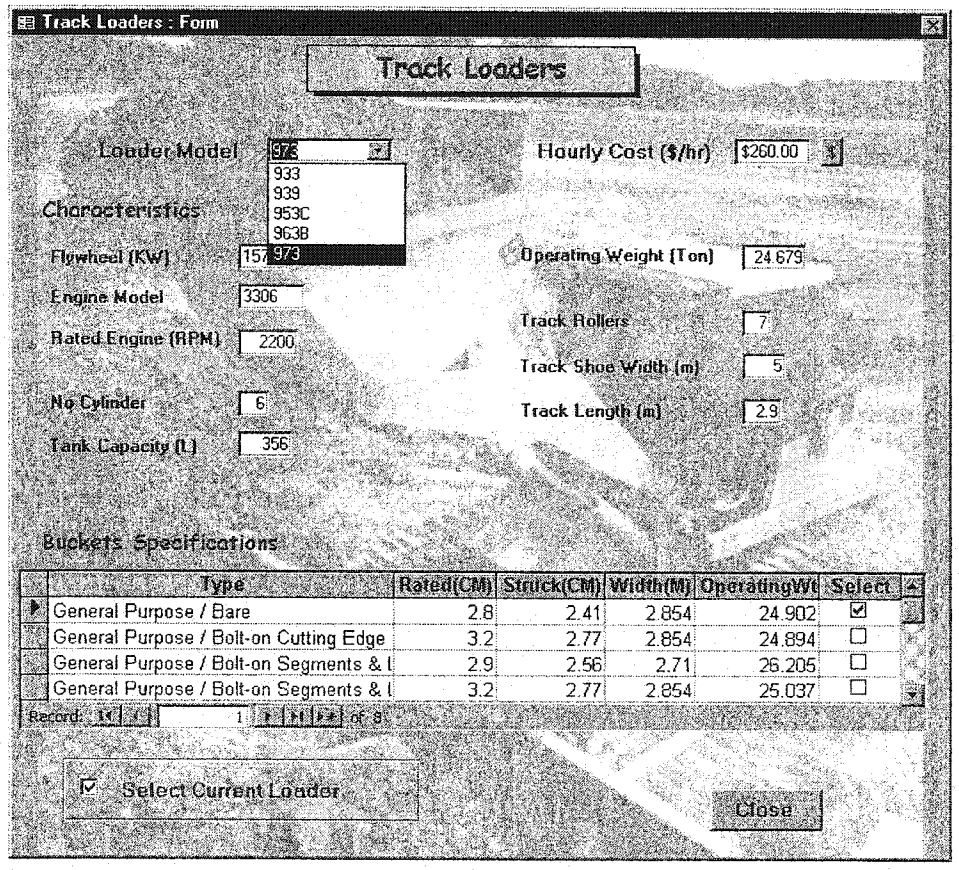


Figure 4.24 Track Loaders User Interface

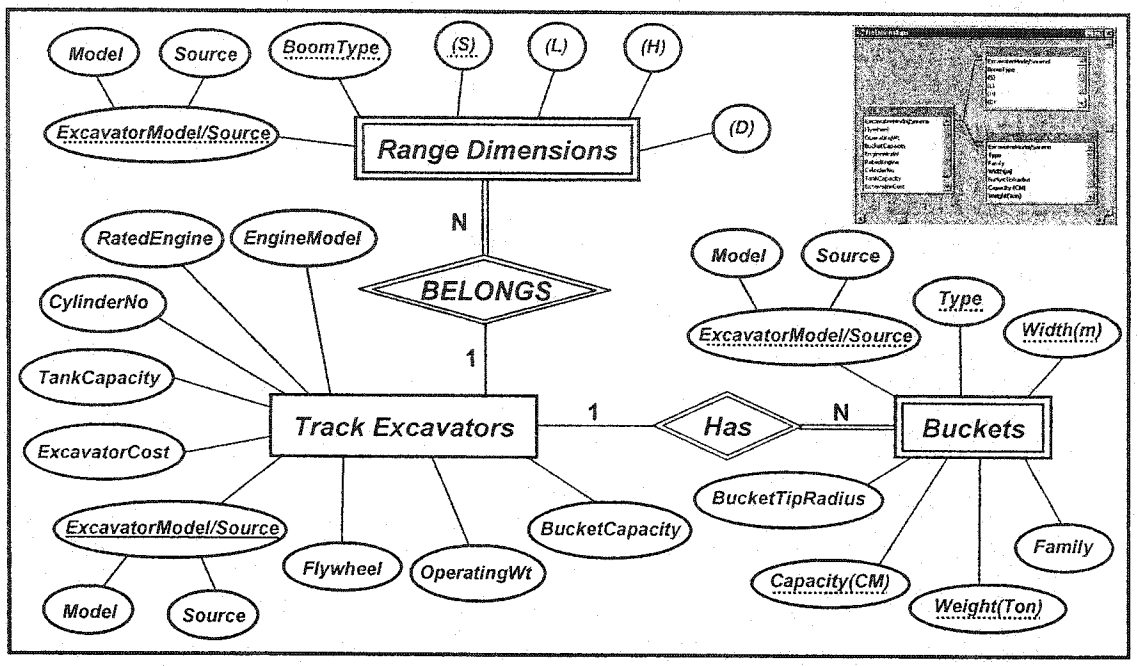


Figure 4.25 ER Diagram for Track Excavators Sub-database

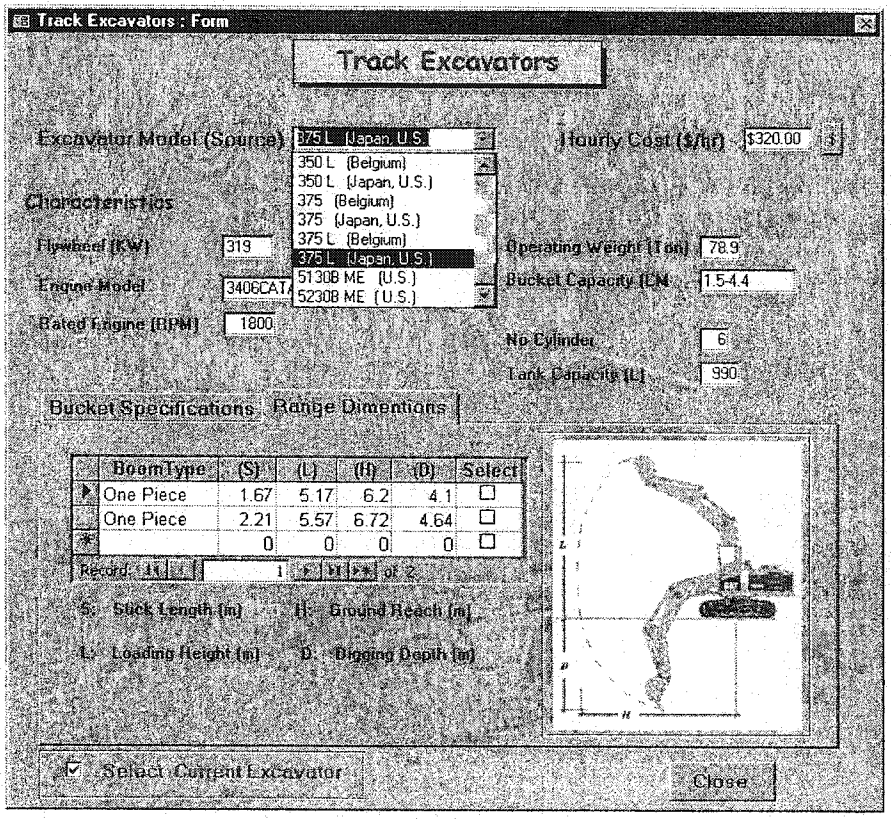


Figure 4.26 Track Excavators User Interface

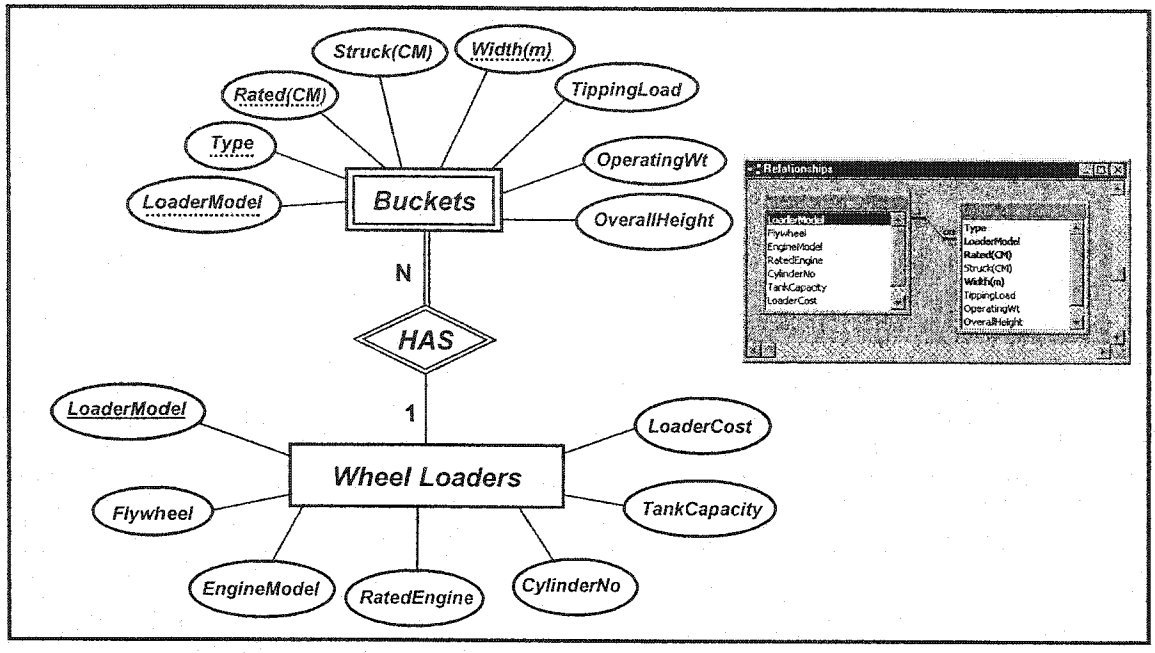


Figure 4.27 ER Diagram for Wheel Loaders Sub-database

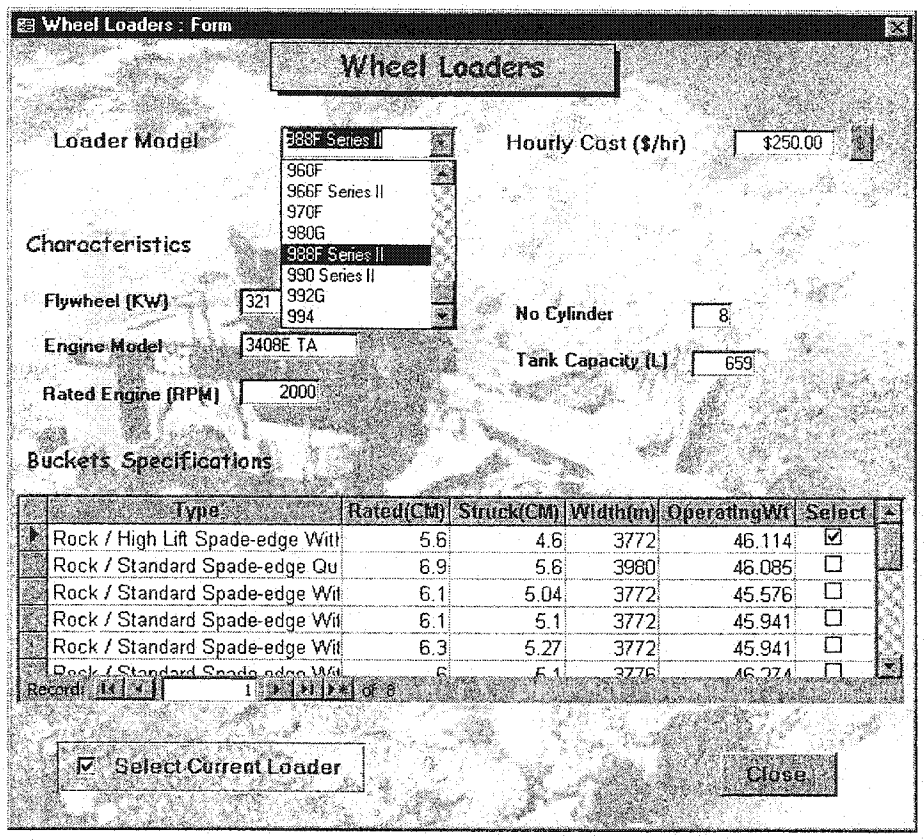


Figure 4.28 Wheel Loaders User Interface

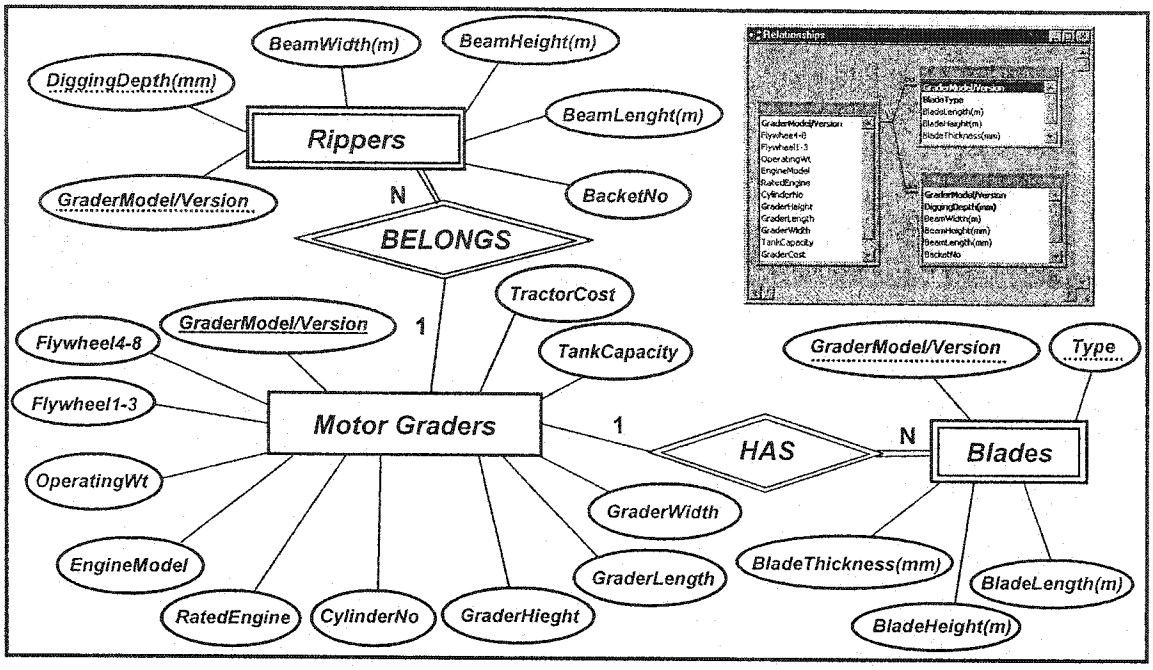


Figure 4.29 ER Diagram for Motor Graders Sub-database

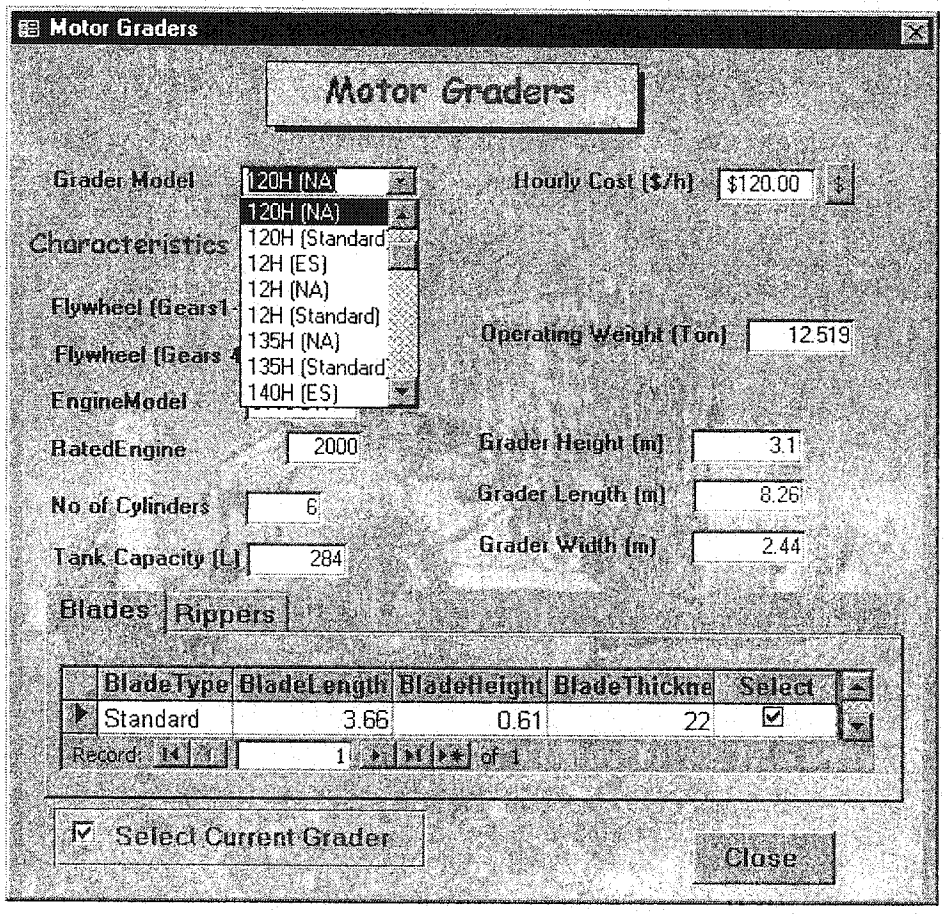


Figure 4.30 Motor Graders User Interface

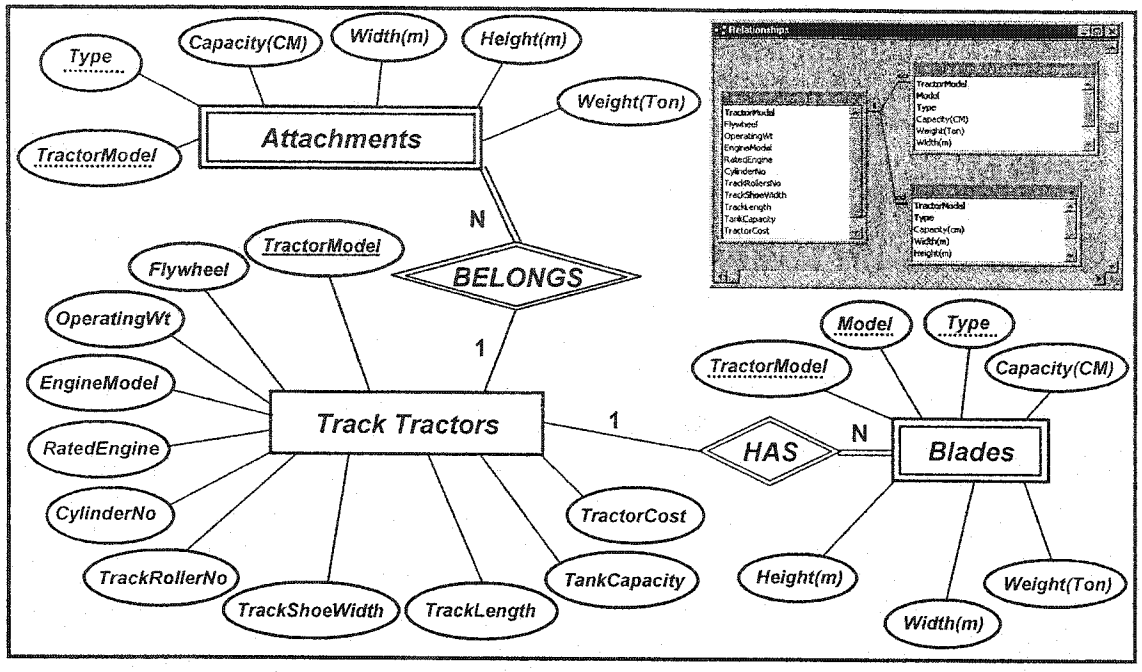


Figure 4.31 ER Diagram for Track-Type Tractors Sub-database

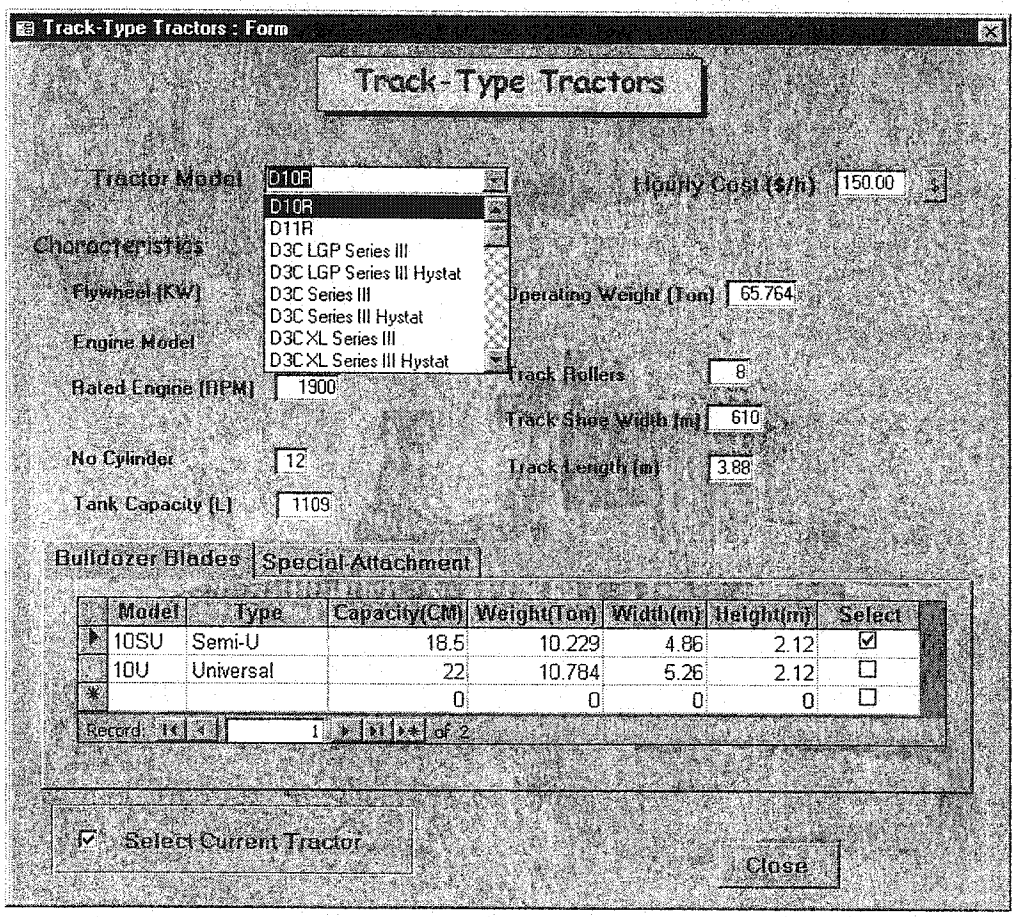


Figure 4.32 Track-Type Tractors User Interface

4.3 Equipment Cost Application

Equipment cost application (*ECA*) has been developed for estimating equipment cost, accounting for all components of equipment costs. *ECA* is essentially a spreadsheet application cost (Moselhi and Marzouk, 2000), designed to provide the user with the total hourly owning and operating costs along with their respective breakdown. In order to estimate equipment owning and operating costs, the calculations are carried out in a highly interactive user friendly environment supported with a set of functions as per the Caterpillar Performance Handbook (1997).

ECA provides further assistance to the user via a set of supporting data tables which can easily be retrieved to support interim calculations by triggering the corresponding button. Upon calculating the owning and operating cost items and entering operator hourly wage, by activating the *Preview* button, *ECA* provides the user with a report that contains hourly cost breakdown for the equipment under consideration.

ECA has a number of interesting features: 1) it provides an easy to use and understand user interface, 2) it allows data import and export to support the integration with other components of the system, 3) it is compatible with the current industry practice, with respect to the cost break down structure and software systems developed by major equipment suppliers. The application was developed in MS Excel and coded using Visual Basic for application (VBA). In the following subsections, a review of equipment cost components which

have been considered in the development *ECA* are presented. A numerical example is also presented in order to demonstrate the practical features of *ECA*.

4.3.1 Equipment Cost Components

Different factors affect equipment cost components including: 1) working conditions; 2) economic conditions; 3) local prices of oil and lubricants; and 4) local wages for equipment operators. Equipment cost components can be grouped into three main categories (Caterpillar 1997): 1) owning costs; 2) operating costs; and 3) wages of equipment operators. Figure 4.33 outlines the three cost categories along with their components. It should be noted that the methodology for estimating equipment hourly owing and operating costs is applicable for any currency considering the exchange rate with respect to $US for undercarriage and repair costs which require the use of multiplier factors as outlined in Caterpillar performance handbook (1997).

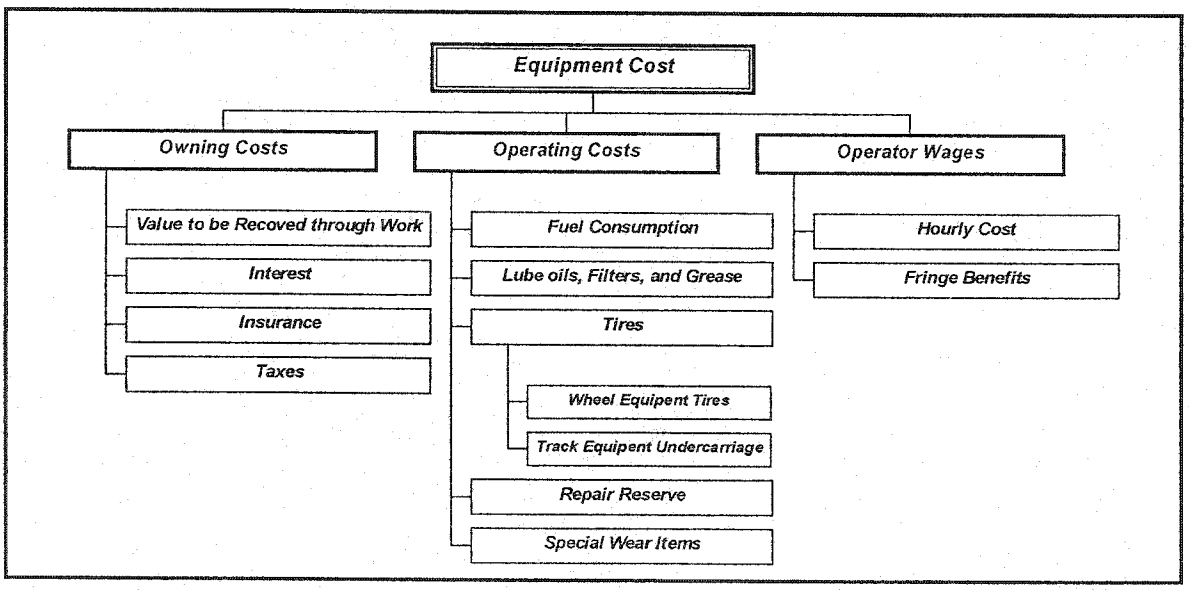


Figure 4.33 Equipment Cost Components (adopted form Caterpillar 1997)

"*Owning Costs*" are the costs of keeping the equipment whether it is used or not (Peurifoy et al 1996). The costs cover the equipment depreciation, interest, insurance and taxes.

- *Value to be Recovered through Work* costs are considered to cover the loss in equipment value (difference between its delivered price and its salvage value). Usually, the price of tires are deducted from the delivered cost since they are considered as a wear item which is included in the operating cost. Different methods are followed to estimate depreciation cost including (Peurifoy et al 1996): 1) straight-line method; 2) declining balance method; and 3) sum of the years (SOY) method. *ECA* calculates the hourly depreciation cost based on the straight-line method according to the following equation (Caterpillar 1997):

$$\text{Depreciation Cost} = \frac{\text{Delivered Price - Salvage Value}}{\text{Ownership Periods (Years)} \times \text{Estimated Usage (Hours/Year)}} \quad (4.1)$$

Caterpillar Performance Handbook (1997) recommends the estimated usage hours for different equipment, considering operating conditions. *ECA* stores these values to be retrieved for use as input data.

- *Interest* costs are considered to cover the cost of using the capital paid on the equipment. The cost of the capital accounts for different factors, including: 1) time value of the money and 2) interest on loans. *ECA* calculates interest cost as follows (Caterpillar 1997):

$$\text{Interest Cost} = \frac{\left[\frac{N+1}{N} \times \text{Delivered Price}\right] \times \text{Interest Rate(\%)}}{\text{Estimated Usage(Hours/year)}} \qquad (4.2)$$

where *N* is the estimated ownership period in years and *Delivered Price* represents the purchasing cost.

- *Insurance and Taxes* costs can be considered, either by: 1) entering their values directly in their corresponding cells (annual insurance or taxes divided by estimated usage), as shown in Figure 4.37, or 2) using insurance and tax rates by substituting in the following two equations (Caterpillar 1997):

$$\text{Insurance Cost(\$/Houre)} = \frac{\left[\frac{N+1}{N} \times \text{Delivered Price}\right] \times \text{Insurance Rate(\%)}}{\text{Estimated Usage(Hours/year)}} \qquad (4.3)$$

$$\text{Tax Cost(\$/Houre)} = \frac{\left[\frac{N+1}{N} \times \text{Delivered Price}\right] \times \text{Tax Rate(\%)}}{\text{Estimated Usage(Hours/year)}} \qquad (4.4)$$

where *N* is the estimated ownership period in years.

"*Operating Costs*" are those costs which result from equipment operation and use. These costs cover the cost of fuel consumption, lubricants, filters, grease, tires (wheel-type equipment), undercarriage (track-type equipment), repair and special wear items.

- *Fuel Consumption* costs are calculated according to the following equation:

Fuel cost($/Houre) = Hourly Fuel Consumption(Units/Hour) × Fuel Unit Price($) (4.5)

Utilizing *ECA*, hourly fuel consumption can be considered, either by: 1) entering their values directly in their corresponding cell (see Figure 4.38) based on field study measurement, or 2) using supporting tables by triggering the corresponding *Fuel* button. These tables provide the hourly fuel consumption based on the load factor which reflects operating conditions operators.

- *Lubrication, Filters and Grease* costs consist of both material cost and hourly labor cost. The user needs to trigger the *GetValue* button (see Figure 4.38) to obtain these costs.

- *Tires* costs are considered only for wheel-type equipment. These costs are affected by different factors including: 1) equipment speeds; 2) regular equipment maintenance; 3) wheel position; and 4) job conditions (surface, curves, grades and loads). Utilizing *ECA*, hourly tire costs can be considered, either by: 1) entering their values directly in their corresponding cell (see Figure 4.39) based on experience, or 2) using the Caterpillar curves by triggering the corresponding *Tires* button (see Figure 4.39). Figure 4.34 depicts one of these curves for construction and mining trucks. These curves were converted into tabular form to ease their use in *ECA*.

- *Undercarriage* costs are considered only for track-type equipment. These costs are affected by different factors, including: 1) impact (i.e. bending or cracking of track structure); 2) abrasiveness (crushing of track components due

to underfoot soil); 3) environmental factors (i.e. existence of chemical on the soil, soil temperature, etc.); 4) operation factors (operators' skills and practices); and 5) maintenance. *ECA* follows the Caterpillar's approach in the estimation of hourly undercarriage costs. In this approach, each piece of equipment has its own basic factor which is multiplied by the summation of impact, abrasiveness and the "Z" factor (combination of environmental, operation, maintenance effects).

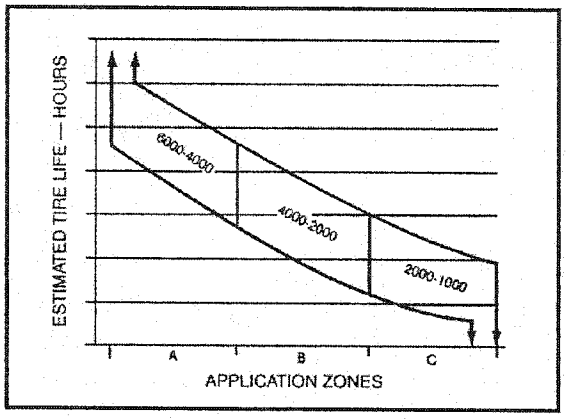


Figure 4.34 Estimated Tire Life Curve (Construction and Mining Trucks, Caterpillar 1997)

- *Repair* costs depend on equipment age. They start low and proportionally increase with equipment age. Different factors influence repair costs including equipment application, operating condition and maintenance plan. Utilizing *ECA*, hourly repair costs can be considered, either by: 1) entering their values directly in their corresponding cell (see Figure 4.40) based on adequate records, or 2) using the Caterpillar approach by triggering the corresponding *GetValues* button (see Figure 4.40). In this approach, hourly repair costs are obtained by

multiplying the equipment basic factor by its extended-life multiplier. The basic factor depends on the operating conditions of the equipment, whereas, the extended-life multiplier is based on the estimated equipment's usage hours. *ECA* stores basic factors' charts in tabular forms to ease their use. Figure 4.35 depicts one of these charts for track-type tractors equipment.

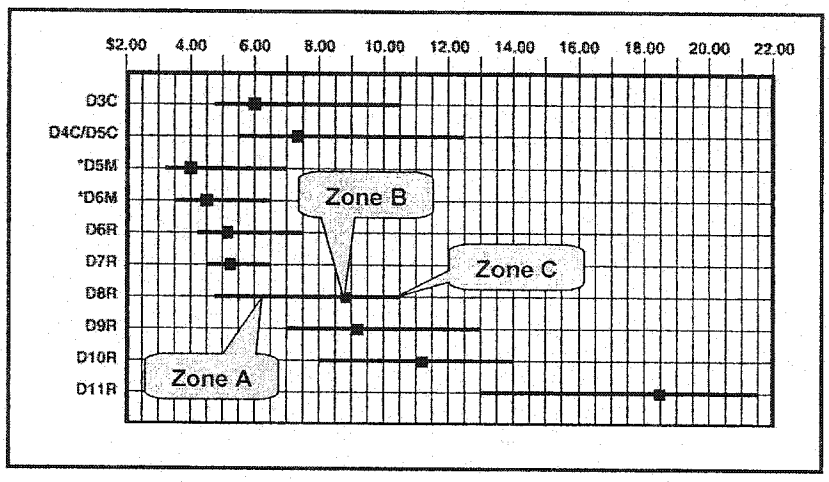


Figure 4.35 Equipment Basic Factor (Track-Type Tractor, Caterpillar 1997)

- *Special Wear Items* costs are application-dependent which vary according to soil and operation conditions. These costs include cutting edges, ripper tips, bucket teeth, etc. *ECA* calculates the hourly cost by dividing the cost of each item by its estimated life in hours.

"*Operator Wages*" are based on operator's hourly cost and his/her fringe benefits. They vary according to the nature and location of the project (i.e. they differ from one place to another) and the contract under consideration (whether covered by a union contract or construction decree).

4.3.2 Numerical Example

In order to demonstrate the practical features of the developed *ECA* and illustrate its capabilities, a numerical example is presented. In this example, an earthmoving contractor is required to estimate the hourly owning and operating cost of a front shovel, which has the following characteristics:

- Purchasing (delivered) price with attachments is $2,520,000
- The shovel will be used for three years with approximately 10,000 hrs ownership period
- Salvage value at the end of use period represents 30% of the delivered price
- Interest Rate = 10%
- Insurance Rate = 1%
- Tax Rate = 1%
- Fuel unit price = $0.45/Litre (Load factor is high)
- The condition multiplier is moderate for the impact, abrasiveness and "Z" factor
- The operating condition is Zone C
- Special wear items estimated to cost $5,000 for an estimated life of 200 hrs
- Operators hourly wage including fringe benefits is $37.5

Step 1:

The user selects, in an interactive manner, the machine type from a menu list as shown in Figure 4.36. Then, he/she enters its ownership period in years and ownership usage (the total operating hours during its ownership period). The user can also obtain ownership usage periods utilizing those incorporated in the system by activating the *GetValue* button (see Figure 4.36) as specified by equipment manufacturers.

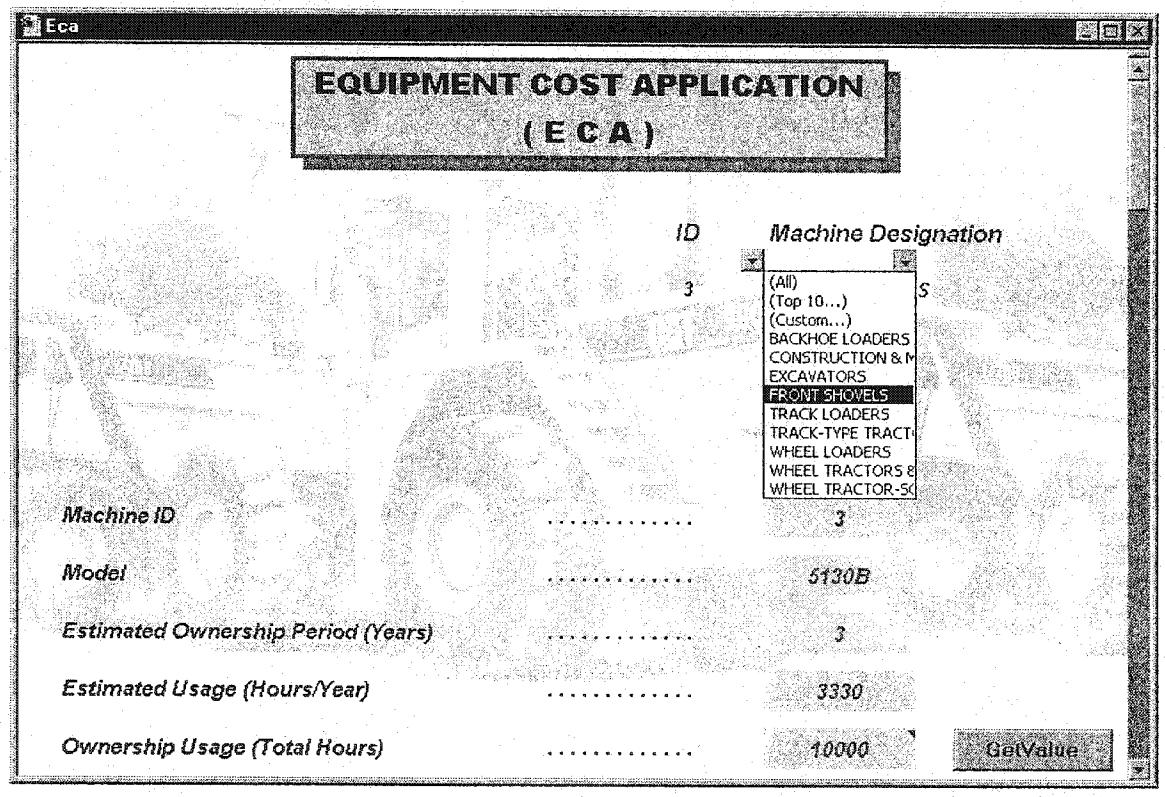


Figure 4.36 Machine Characteristics User Interface

Step 2:

The user is required to enter owning costs data, i.e. the delivered price, salvage value percentage, interest rate, insurance rate, and tax rate. Finally he/she ends up with the hourly owning costs which equal $236.94. Figure 4.37 depicts the user interface screen for owning costs.

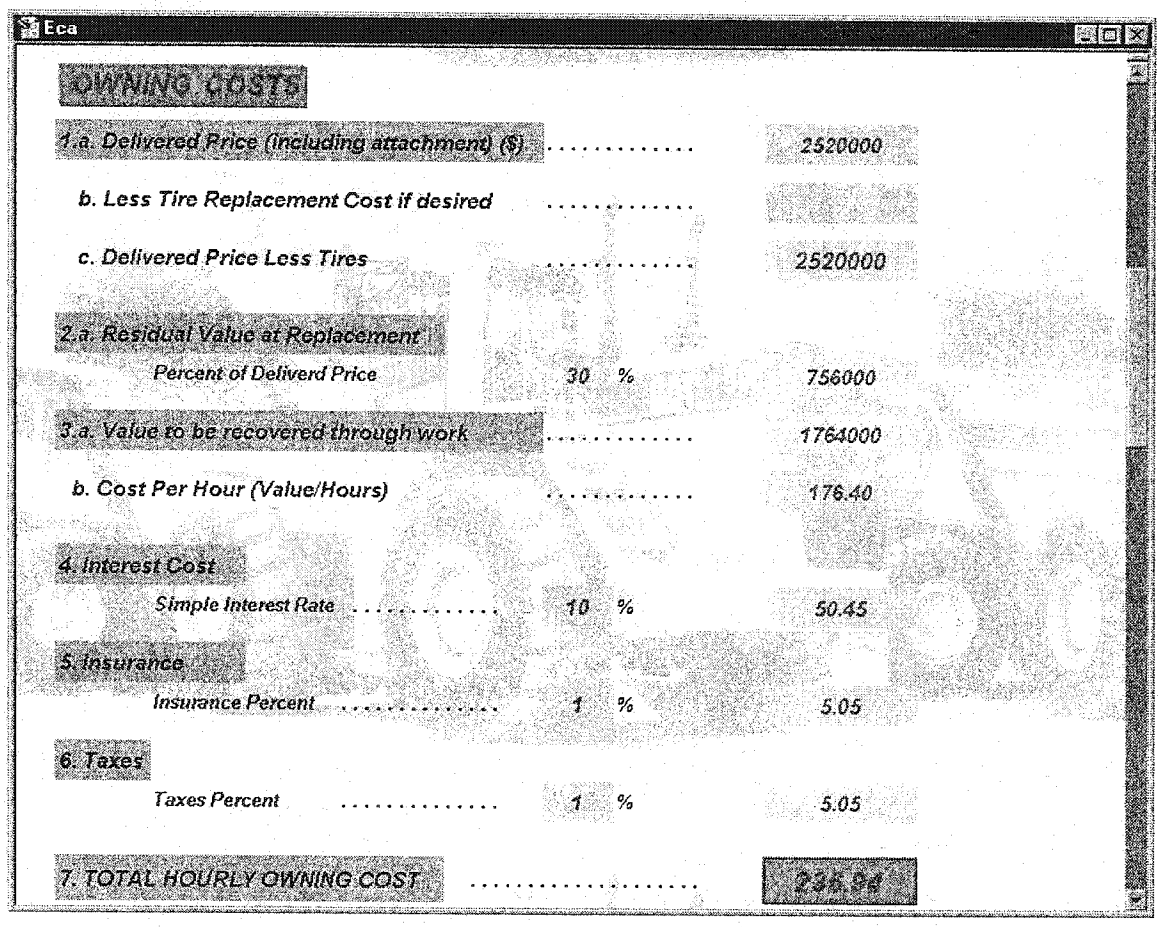


Figure 4.37 Owning Cost User Interface

Step 3:

The user is required to enter operating costs data (i.e. the fuel unit price, consumption rate of fuel, the hourly cost of the lube oils, filters, and grease, multipliers of undercarriage and repair, basic factor of undercarriage and repair, costs of special items, and their estimated life).

First: the consumption rate is obtained by triggering the *Fuel* as shown in Figure 4.38. *ECA* provides the user with the consumption rates in liters and gallons for different loading factors (low, medium and high).

Second: the user triggers the *GetValue* button to obtain the hourly cost for lube oils, filters, and grease for the given equipment. It provides both material and labor costs (see Figure 4.38).

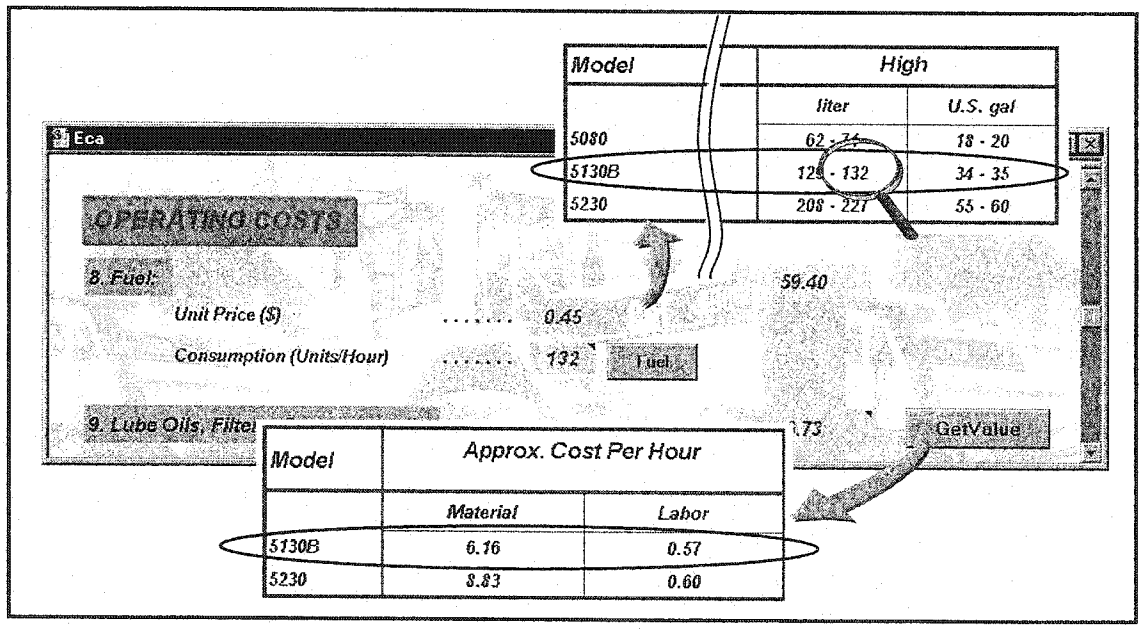


Figure 4.38 Fuel and Lube Oils Costs User Interface

Third: undercarriage costs are calculated utilizing both condition multipliers and basic factors. For the given machine, should the user try to enter data related to tires, he/she will receive a warning message. By activating the *GetValues* button, *ECA* provides the user with the undercarriage multipliers for impact, abrasiveness and "Z" factor in different operating conditions (high, moderate and low) along with the basic factor of the machine (see Figure 4.39).

Fourth: repair reserve costs are calculated by entering both the extended use multiplier and the basic repair factor. By activating the *GetValues* button, *ECA* lists all the extended use multiplier according to the estimated usage hours of the equipment. It also provides the user with the basic repair factor in different zones (A, B and C) (see Figure 4.40).

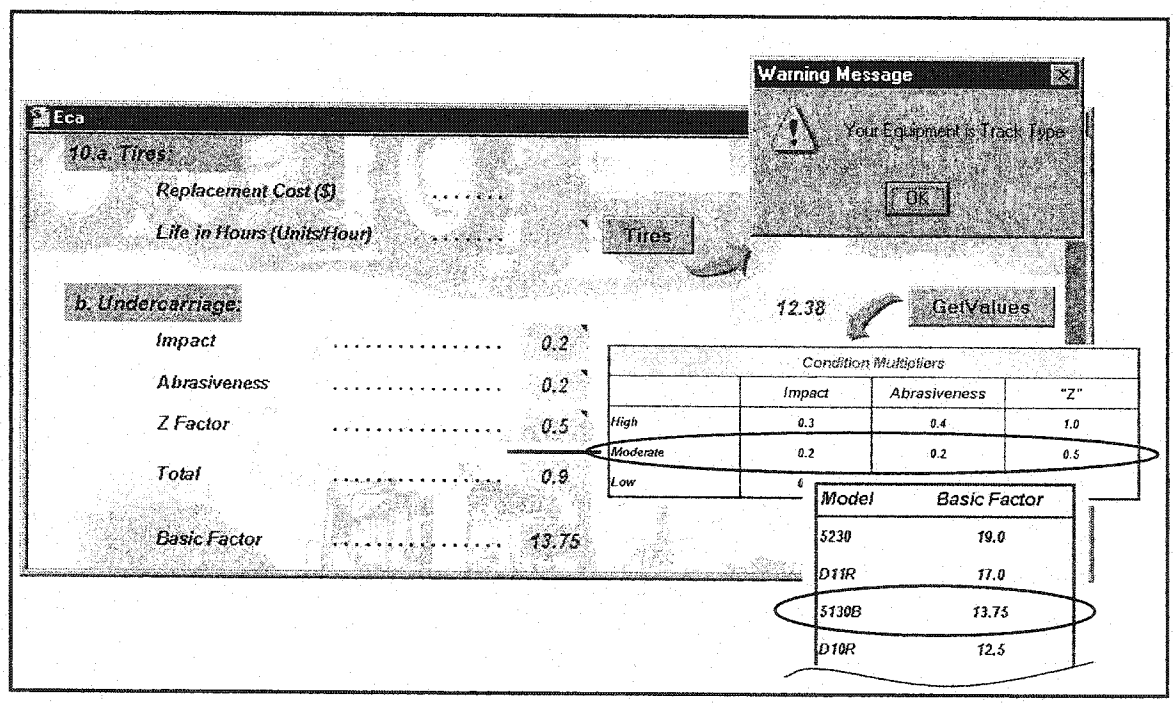


Figure 4.39 Undercarriage Costs User Interface

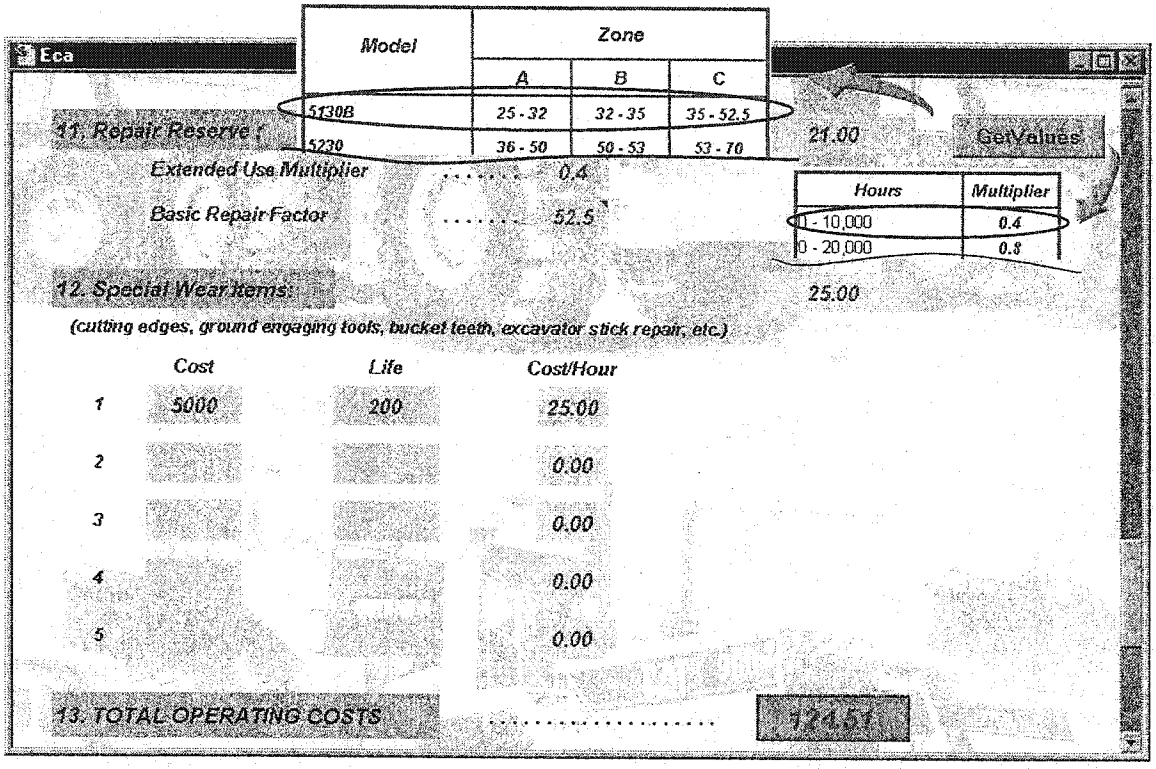


Figure 4.40 Repair and Special Wear Items Costs User Interface

Fifth: special wear items costs are calculated by entering the initial cost and the expected life time for each item as illustrated in Figure 4.40.

Finally: the user ends up with the hourly operating costs which equal $124.51 (see Figure 4.40).

Step 4:

The user enters the operator hourly wage including the fringe benefits in the corresponding cell (see Figure 4.41). Subsequently, *ECA* calculates the total owning and operating costs for the given shovel which equal $398.95 as per

Figure 4.41. By activating the *Preview* button, *ECA* provides the user with a report that contains hourly cost breakdown for the equipment under consideration which can be printed by triggering the *Print* button (see Appendix B).

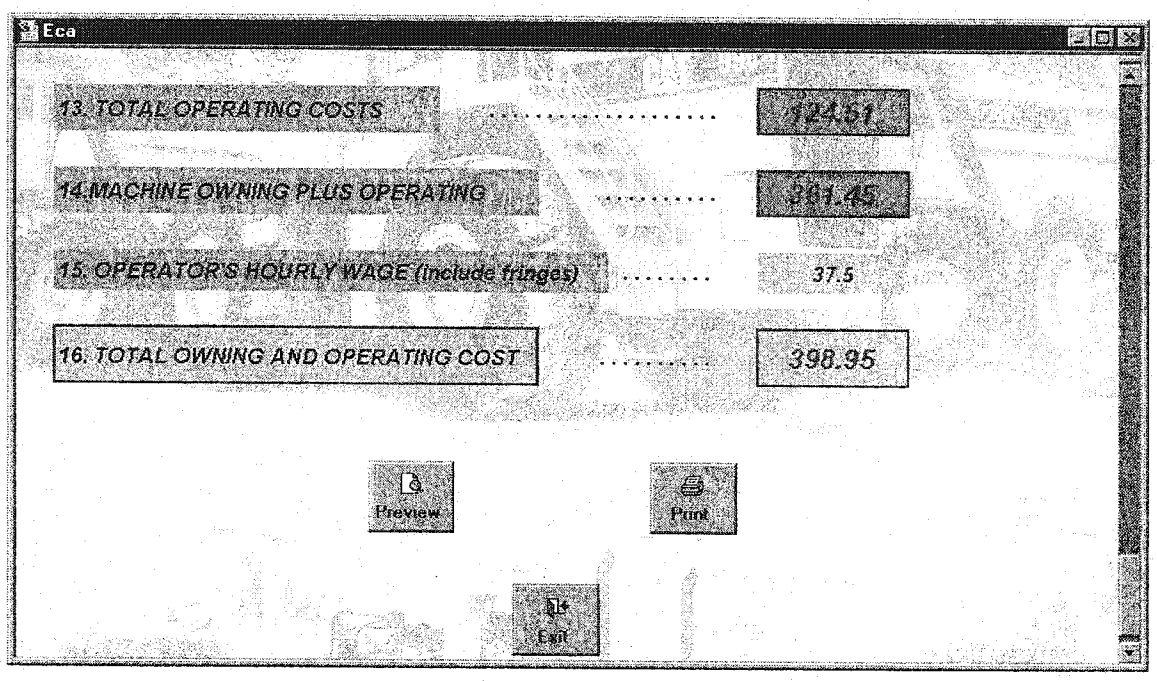


Figure 4.41 Total Owning and Operating Costs User Interface

4.4 Summary

Two components of the proposed *SimEarth* system have been presented in this chapter: 1) earthmoving equipment database (*EDA*) and 2) equipment cost application (*ECA*). Also presented is a review of the different database models. The characteristics of *EDA* and retrieval of equipment data (e.g. equipment characteristics, equipment attachments and equipment range dimensions) have

been described. The design process of *EDA* has also been described along with its implementation in MS Access 97, benefiting from its powerful user interfaces and reporting capabilities.

The main categories of equipment cost along with their components have been reviewed. The chapter also presented the development of equipment cost application (*ECA)*. The application has been designed to assist earthmoving contractors in estimating the hourly owning and operating costs for each piece of equipment in their fleet. The most interesting features of *ECA* have been highlighted along with different interactive user interfaces and automated data retrieval. A numerical example has been presented to demonstrate the essential features of *ECA* and its practical use.

CHAPTER 5

FUZZY CLUSTERING AND MARKUP APPLICATIONS

5.1 General

This chapter presents the development of two components of *SimEarth*. These are: Hauler's Travel Time Application (*HTTA*) and EarthMoving Markup Application (*EMMA*). *HTTA* is a fuzzy clustering method for estimating haulers' travel time. The proposed method utilizes linear regression and fuzzy subtractive clustering. Seven factors influencing haulers' travel time were first identified and their significance was then quantified using linear regression. The regression analysis was performed utilizing 180 training cases, generated using commercially available software for different models of haulers. The data were generated randomly to represent a wide range of possible combinations of factors affecting travel time of haulers across different types of road segments. The training data was subsequently used in the development of the proposed method. A numerical example is presented to demonstrate the use of the developed method and to illustrate its accuracy.

EMMA provides a decision support environment for estimating markup, essential for successful bid proposals. The developed model can assist earthmoving contractors in estimating markups and owners and/or their agents in evaluating bid proposals. The model is generic and is not limited only to earthmoving. In

addition, it can be used as a tool to evaluate different alternatives in engineering, procurement and construction. It combines the advantages of multi-attribute utility theory and the analytic hierarchy process. Unlike models developed for similar purposes, the proposed model provides a decision support environment for the two functions, i.e. estimation of markup and evaluation of bids. It also enables the user to construct the decision hierarchy that best suits his/her company's business environment and bidding strategy in a flexible manner. It accounts for the decision-maker's attitude towards risk. The model is coded using MS visual basic 6.0 (providing a user-friendly interface to facilitate its use) and is integrated with MS Excel to allow for easy data import and export. A numerical example is presented to demonstrate the use and capabilities of the proposed model.

5.2 Fuzzy Clustering Application

5.2.1 Background

Fuzzy logic was introduced to work as an inference that resembles human reasoning capabilities in the form of knowledge-based systems (Fuller 2000). These systems have proven to be good modeling tools for complex problems where mathematical solutions are difficult to formulate (Fuller 2000). Cluster algorithms (self-organizing map) are used to group data into subsets or clusters that contain data having similar feature(s). Different cluster algorithms have been developed in various applications using fuzzy logic (Bezdek and Pal 1992, Yager

and Filev 1994, Chiu 1994). In those algorithms, data is expressed in a form of fuzzy rules, each representing a cluster. Those algorithms include: 1) fuzzy C-Means (FCM) clustering method (Bezdek et al 1987), 2) mountain clustering method (Yager and Filev 1994) and 3) subtractive clustering method (Chiu 1994).

FCM clustering method (Bezdek et al 1987) is an iterative technique that starts with a set of cluster centers and generates membership grades, used to induce new cluster centers. The number of iterations depends on the choice of the initial values of the clusters' centers. Mountain clustering method (Yager and Filev 1994) is based on creating a grid of data space and computing the potential value (mountain function) for each point on the grid, based on its distance to the actual data point (see Figure 5.1-a). The greatest potential point (one of the grid vertices) represents the first cluster (highest point on the mountain) as shown in Figure 5.1-b. Subsequently, the potential for each grid point is adjusted, allowing for the determination of all remaining clusters.

Subtractive clustering method (Chiu 1994) is an extension of the mountain clustering method, where the potential is calculated for the data rather than the grid points. As a result, clusters are elected from the system training data according to their potential as depicted in Figure 5.2. Subtractive clustering has an advantage over mountain clustering in that there is no need for estimating a resolution for the grid.

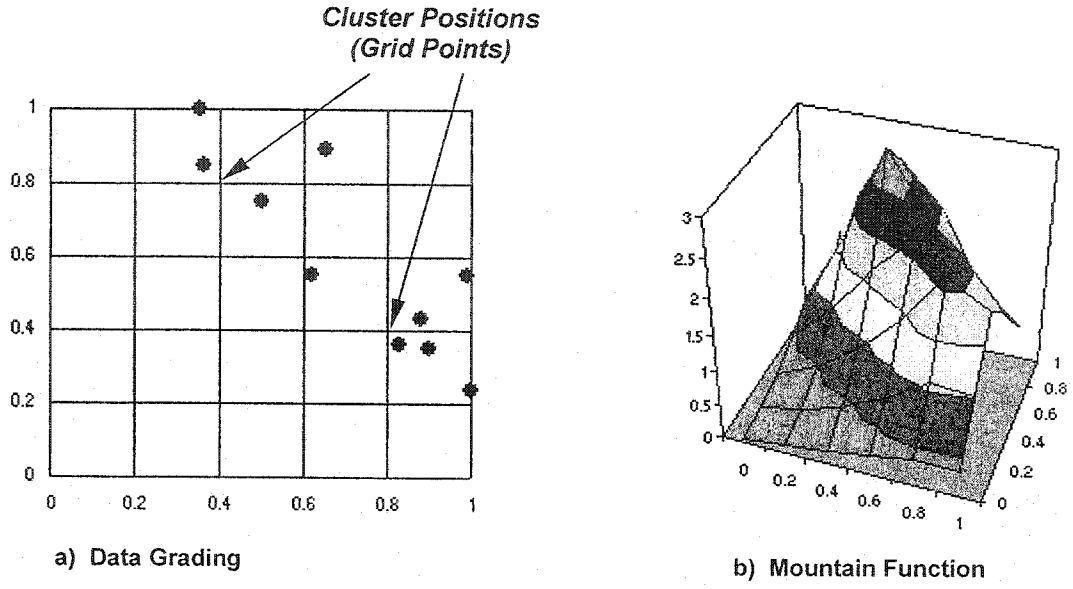


Figure 5.1 Mountain Clustering Method (Yager and Filev 1994)

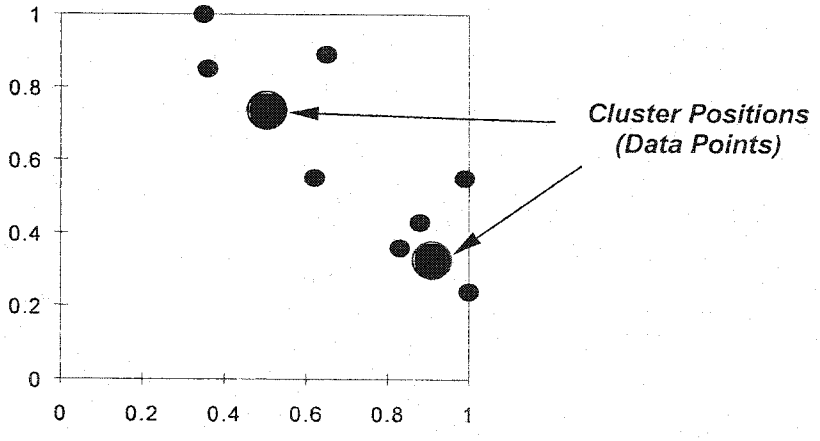


Figure 5.2 Subtractive Clustering Method (Chiu 1994)

5.2.2 Haulers' Travel Time

Accurate calculation of travel time is essential for determining productivity of earthmoving operations, and for ultimately selecting the most cost effective fleet configuration for executing the work. Factors influencing the performance of haulers can be grouped as: 1) those related to the condition of haulers themselves and 2) those related to the characteristics of haul roads. The first category has to do with the age of the equipment and the wear and tear resulting from its use over time. This category is not included in the developed method. It is assumed that proper maintenance of equipment during project execution will minimize its impact on travel time. The second category impacts the performance of earthmoving haulers, regardless of their conditions, and directly affects their travel time. This category will be considered in the proposed method.

Manufacturers' performance charts and/or site-collected data are commonly used to estimate haulers' travel time. These charts provide the hauler speed under positive and negative total resistances without accounting for acceleration and deceleration zones. These charts are called *Rimpull-Speed-Gradeability* and *Brake Performance*, respectively (Caterpillar 1997). Further, charts known as *Travel Time* charts, provides haulers' travel time under loaded and unloaded conditions (Caterpillar 1997). These charts, however, cannot be directly used for haul profiles having road segments with different total resistance.

Hicks (1993) proposed a performance equation for determining haulers' speeds,

and hence their travel time. The equation was developed using regression analysis. Although the equation is helpful, compared to using performance charts, it does not account for acceleration and deceleration in transition zones. Caterpillar developed commercially available software (FPC 1998), dedicated for estimating productivity of equipment fleets used in earthmoving operations in a deterministic manner. Although FPC estimates haulers' travel time considering acceleration and deceleration in transition zones, it lacks the portability and availability necessary for use by researchers. It cannot be incorporated in simulation models that are developed by others. Conversely, the method proposed in this chapter, does not improve the estimated travel time over that obtained from FPC, rather, it provides a generic tool that can be incorporated in models dedicated for estimating earthmoving production. As such, it provides an improvement in the accuracy of estimating haulers' travel time over those methods used in recent simulation models dedicated for earthmoving operations (Simphony 2000). The developed method, further, facilitates data entry with respect to the maximum allowable speed of haulers traveling across road segments having negative total resistance. This was accomplished by generating a table entry in a database designated for the maximum allowable speed over a wide range of haulers models, accounting for load percent of haulers and total resistance of the road segments. This feature does not exist in FPC software.

The following sections present a two-step fuzzy clustering method for estimating haulers' travel time, accounting for: 1) the performance of haulers over their life

time and 2) acceleration and deceleration in transition zones. In the first step, the linear regression technique (Neter and Wasserman 1974, Keller and Warrack 1998) is performed in order to identify factors that significantly affect travel time. In the second step, fuzzy clustering is used for modeling haulers' travel time. Three subtractive clustering models have been tested and evaluated for possible use in the developed method. These are: un-optimized (UO) subtractive clustering, optimized Takagi-Sugeno 0^{th} order (OTS_0) subtractive clustering and optimized Takagi-Sugeno 1^{st} order (OTS_1) subtractive clustering. The OTS_1 model out performed the others and was accordingly utilized in the development of the proposed method. A set of thirty-six randomly generated cases, representing travel of haulers across different road segments, has been analyzed to demonstrate the accuracy of the developed method. The proposed method has a number of attractive features: 1) it is accurate and easy to use; 2) it can be easily incorporated into other earthmoving modeling techniques regardless of whether they are based on simulation (Marzouk and Moselhi 2000, AbouRizk and Hajjar 1998), queuing theory (Halpin and Woodhead 1976), and/or regression (Smith 1999); and 3) it overcomes some of the limitations of the commercially available software FPC. The method has been implemented utilizing MS visual basic for application (VBA) macros in MS Excel 97 environment.

5.2.3 Model Parameters

Seven factors have been identified from the literature to impact haulers' travel

time. These are: 1) the maximum allowable speed in the current road segment; 2) the maximum allowable speed in the previous road segment (if the current is not the first one) which may cause the hauler to accelerate; 3) the maximum allowable speed in the next road segment (if the current is not the last one) which

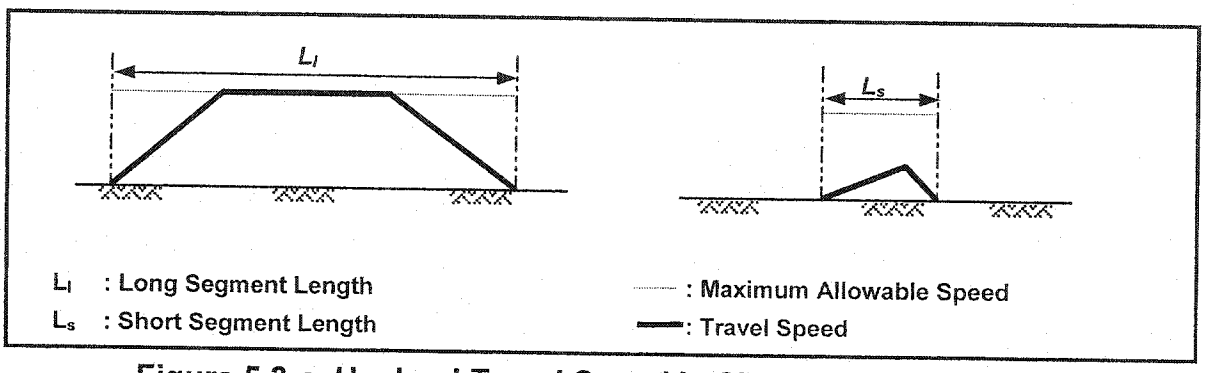


Figure 5.3-a Haulers' Travel Speed in Single Road Segments

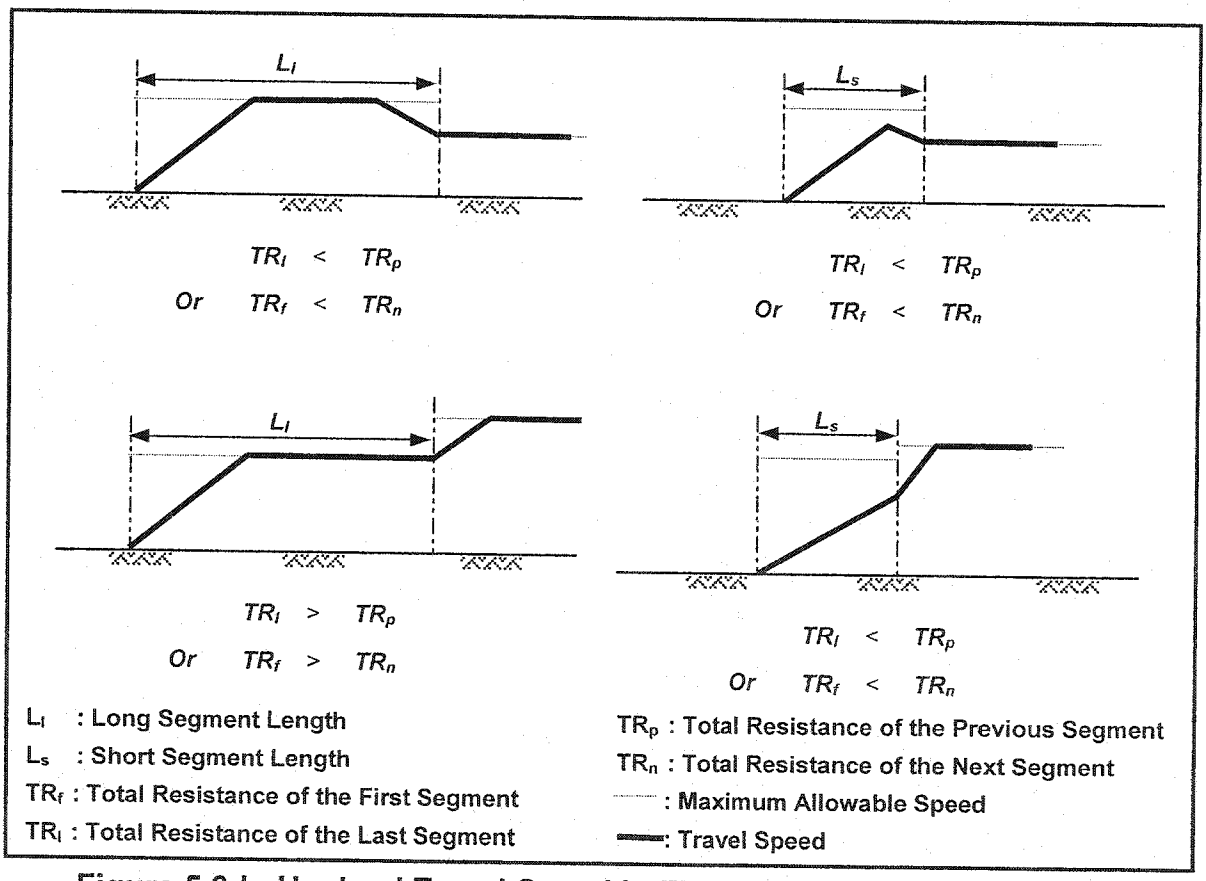


Figure 5.3-b Haulers' Travel Speed in First and Last Road Segments

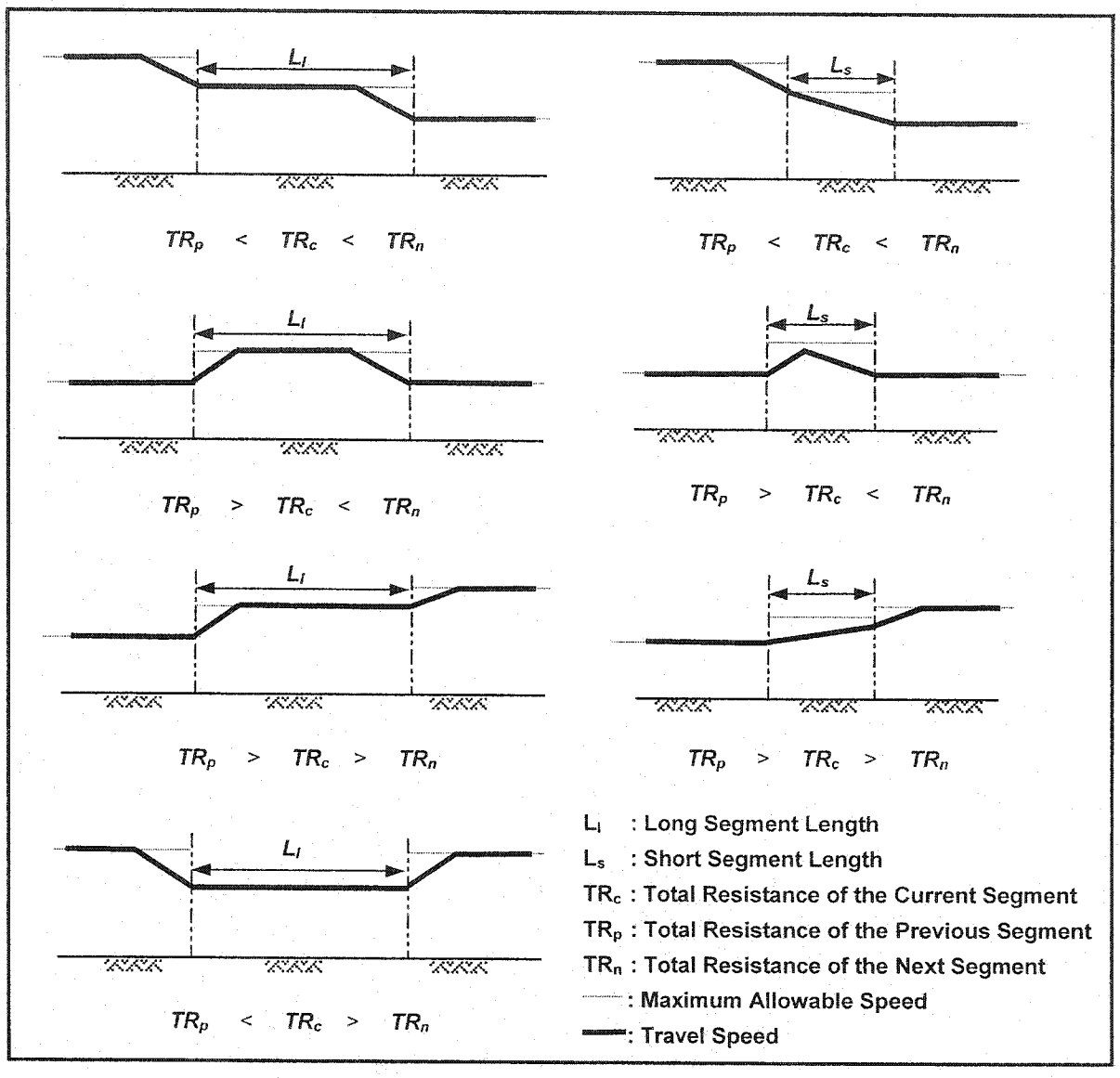


Figure 5.3-c Haulers' Travel Speed in Intermediate Road Segments

may also cause the hauler to decelerate; 4) the length of road segment which may (in case of long length) or may not (in case of short length) allow the hauler to reach its maximum allowable speed; 5) the total resistance which combines the segment grade and its rolling resistance, expressed in percent; 6) hauler load percent (ratio of the hauler's load to its maximum capacity) and 7) hauler's

model. Figures 5.3-a, b and c depict possible scenarios for haulers' speeds across single, first or last, and intermediate road segments. In order to account for the acceleration and deceleration of a hauler in modeling its average speed, the seven factors, stated earlier, are considered in this study.

The cases used in developing and testing the proposed method have been generated using the FPC software (1998). In generating these cases, a limit was set on the haulers' allowable speed, in downhill cases, in order to make it compatible with the characteristics of the hauler under consideration. The limiting values for the haulers considered in the development of the proposed method were obtained from the retarder charts of the respective hauler's model (Caterpillar 1997). Data for input parameters such as segment length, total resistance and hauler's percent load have been randomly generated to capture a wide range of possible conditions as depicted in Figures 5.3-a, b, and c. The cases were grouped into two categories: 1) training cases (180 sets) for the development of a fuzzy subtractive model and 2) testing cases (36 sets) to validate the developed model. The cases were generated considering the input variables and their respective ranges shown in Table 5.1. It should be noted that for projects planned to be executed in different phases, the characteristics of the road segments remain constant during the execution of each individual phase, but can vary from one phase to another. This is a reasonable assumption considering that road segments are usually maintained, as good practice, to achieve good overall efficiency within each phase.

Table 5.1 Limits of Input Variables

Input Variable	*Unit*	*Upper Limit*	*Lower Limit*
Length	*meter*	*5000*	*10*
TR	*%*	*20*	*-20*
Load Percent	*%*	*100*	*75*

5.2.4 Regression Model

Regression was performed in two steps: 1) multiple regression, to evaluate the impact of the hauler's model, being a qualitative factor and 2) stepwise regression, to assess the significance of the quantitative factors on haulers' travel time. In the multiple regression step, training and testing data have been generated for six off-highway trucks: models 769D, 773D, 777d, 785C, 789C and 793C. The model of the hauler has been considered as a qualitative dummy variable (Keller and Warrack 1998). In this study, the dummy variable takes either 1 or 0 value. If 1 is assigned to a hauler's model, that model is considered in the analysis, otherwise it is not. For example, the shaded row in Table 5.2 indicates that the hauler model 789C has been considered along with the quantitative variables impacting haulers' travel time across that particular road segment. As a result, the final regression model consists of 12 variables. The results of the multiple regression analysis are shown in Table 5.3. It should be noted that the coefficients associated with the dummy variables are almost equal, indicating that the hauler's model has insignificant effect on the haulers' average speed.

Table 5.2 Variables of Multiple Regression Analysis

F_769	F_773	F_777	F_785	F_789	F_793	PrvSp	NextSp	Length	TR	SegSp	LoadPer	AvSp
0	1	0	0	0	0	64.87	25.54	680	15	9.9	0.96	12.14
0	0	0	0	1	0	54.74	54.74	1,713	18	7.1	0.87	7.46
						.			.	.		
						.			.	.		
0	0	1	0	0	0	9.66	49.37	974	-18	10.76	0.89	10.76

Table 5.3 Results of Regression Analysis

Variable	Coefficient	
Intercept	-250.86	
F_769	247.86	*Dummy Variables*
F_773	248.15	
F_777	247.57	
F_785	246.89	
F_789	247.53	
F_793	247.98	
Previous Segment Allowable Speed	0.0256	
Next Segment Allowable Speed	0.0128	
Segment Length	0.00062	
Total Resistance	-0.00776	
Segment Allowable Speed	0.88102	
Load Percent	3.1908	

Subsequently, stepwise regression has been performed for the different types of road segments. For an intermediate road segment, the six parameters referred to in the previous section are used in the regression model to determine their relative significance to the haulers' travel time. In this process, a stepwise linear regression analysis was performed to avoid multicollinearity (Neter and Wasserman 1974, Keller and Warrack 1998). An Excel macro (Keller and Warrack 1998) was utilized to perform the analysis. The stepwise regression was performed in three steps as depicted in Table 5.4. The procedure, used by the macro, is based on generating a simple regression model for each variable and including the one which has the largest F-statistic. The F-statistic values which indicates whether the independent variables are linearly related to the dependent variable at a specific level of significance (corresponding to t-statistic in simple regression models).

Table 5.4 Stepwise Regression Statistics

Step No.	*F_Statistic*	*R Square*	*Variable*	*Coefficient*
1	5068.08	0.9829	Intercept	1.7115
			Segment Allowable Speed	0.8873
2	2727.23	0.9842	Intercept	0.4646
			Segment Allowable Speed	0.8835
			Segment Length	0.0006
3	1859.38	0.9846	Intercept	-0.2223
			Segment Allowable Speed	0.8827
			Segment Length	0.0006
			Prev. Segment Allowable Speed	0.0268

Subsequently, the macro checks the performance of the model by adding and/or removing independent variable(s) and comparing the resulting F value against "*F-to-enter*" and "*F-to-remove*" values, respectively. The default values for "*F-to-enter*" and "*F-to-remove*" have been set to 0.05 and 0.1 level of significance, respectively. It should be noted that only three out of the six parameters have been found significant, and were used in developing the proposed method. These parameters are: 1) previous segment allowable speed; 2) segment length and 3) segment allowable speed. It should be noted that the total resistance and load percent variables have been removed from the list of variables considered. This could be attributed to the fact that these two variables are accounted for in the segment's allowable speed. Similarly, a stepwise regression analysis has been performed to identify the significant factors that affect hauler's travel time across single, first and last road segments. Only two factors were found significant: 1) segment length and 2) segment allowable speed.

5.2.5 Fuzzy Computational Algorithm (HTTA)

A fuzzy knowledge-based model was developed using a set of IF...THEN type rules (Marzouk and Moselhi 2001). Upon identification of the significant factors as described above, a set of fuzzy rules was generated by projecting fuzzy clusters into the input space (Chiu 1994). As stated above, the procedure proposed by Chiu (1994) has been utilized in the proposed method as outlined in Figure 5.4. The algorithm performs its computations in five steps as follows:

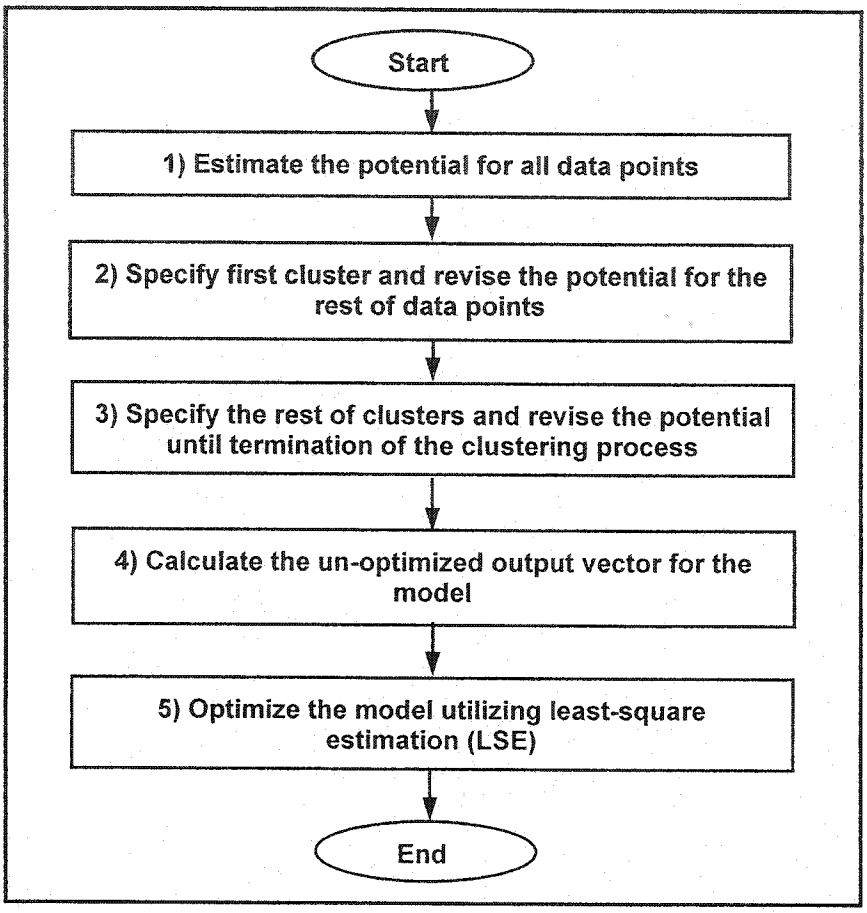


Figure 5.4 Steps of Fuzzy Subtractive Clustering Algorithm

1. Calculate the potential (P_i) for each point in the training data set after normalizing the data in all directions as follows:

$$P_i = \sum_{j=1}^{n} e^{-\alpha \left\| x_i - x_j \right\|^2} \tag{5.1}$$

$$\alpha = \frac{4}{r_a^2} \tag{5.2}$$

In which r_a : radius was that defines the neighborhood (cluster radius), and x_i : data vector for the considered point in both input and output dimension. A computer code has been developed to perform this step (see Appendix A).

2. Set the point of the highest potential P_1^* to be the first cluster of the model. Then, revise the potential for the rest of the data points considering the calculated value of P_1^* as follows:

$$P_i = P_i - P_1^* e^{-\beta \|x_i - x_j\|^2} \tag{5.3}$$

$$\beta = \frac{4}{r_b^2} \tag{5.4}$$

In which r_b : radius introduced to avoid obtaining closely spacing cluster centers. It is greater than r_a and has been recommended to be 1.5 r_a.

3. Repeat the previous step for all remaining clusters in a sequential manner and evaluate them for acceptance/rejection purposes following the procedure outlined in Figure 5.5 by applying the following equation:

$$P_i = P_i - P_k^* e^{-\beta \|x_i - x_j\|^2} \tag{5.5}$$

In which $\underline{\varepsilon}$: threshold for acceptance, $\bar{\varepsilon}$: threshold for rejection, $d_{\min}$: shortest of

the distances between x_k^*, and all previously found cluster centers and *c:* number of clusters.

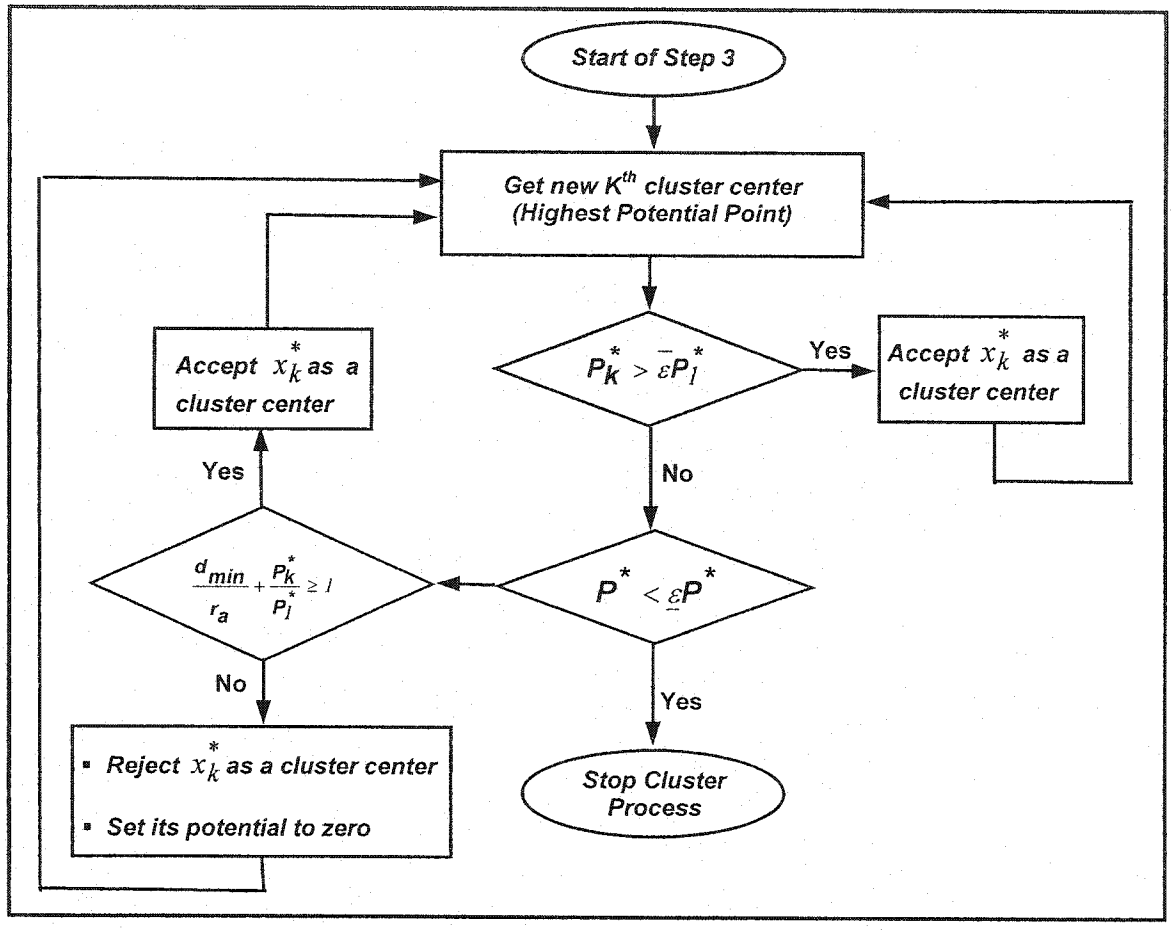


Figure 5.5 Acceptance/Rejection Procedure

4. Calculate the un-optimized output vector Z of the model as follows (Chiu 1994):

$$\mu_i = e^{-\alpha \left\| y - y_i^* \right\|^2} \qquad (5.6)$$

$$Z = \frac{\sum_{i=1}^{c} \mu_i Z_i^*}{\sum_{i=1}^{c} \mu_i} \tag{5.7}$$

In which y_i^*: input vector of cluster i, and z_i^*: output vector of cluster i.

5. Assume a linear output for each cluster ($z_i^* = G_i y + h_i$), the optimization is performed so as to minimize the square error, as reported by Chiu (1994), as follows:

$$Z = \sum_{i=1}^{c} \rho_i (G_i y + h_i) \tag{5.8}$$

$$\rho_i = \frac{\mu_i}{\sum_{j=1}^{c} \mu_i} \tag{5.9}$$

$$\begin{bmatrix} Z_1^T \\ \vdots \\ Z_n^T \end{bmatrix} = \begin{bmatrix} \rho_{1,1} y_1^T & \rho_{1,1} & \cdots & \rho_{c,1} y_1^T & \rho_{c,1} \\ & & \vdots & & \\ \rho_{1,n} y_1^T & \rho_{1,n} & \cdots & \rho_{c,n} y_c^T & \rho_{c,n} \end{bmatrix} \begin{bmatrix} G_1^T \\ h_1^T \\ \vdots \\ G_c^T \\ h_{c1}^T \end{bmatrix} \tag{5.10}$$

In which Z_n^T: output vector of data point *n*, y_n^T: input vector of data point *n*, and G_c^T and h_c^T: parameters of Takagi-Sugeno type polynomial function for cluster *c*.

5.2.6 Modeling of the Average Speed

The parameters found to be significant, using the stepwise regression described earlier, are considered in developing the proposed method. The methodology has been applied for eight scenarios representing: 1) a loaded hauler crossing a single segment (*Haul_Single*), 2) an empty hauler crossing a single segment (*Return_Single*), 3) a loaded hauler crossing the first segment (*Haul_First*), 4) an empty hauler crossing the first segment (*Return_First*), 5) a loaded hauler crossing the last segment (*Haul_Last*), 6) an empty hauler crossing the last segment (*Return_Last*), 7) a loaded hauler crossing an intermediate segment (*Haul_Mid*) and 8) an empty hauler crossing an intermediate segment (*Return_Mid*).

The 180 randomly generated training data cases were normalized to have values between 0 and 1.0. The radius defining the neighborhood for each cluster (r_a) has been determined using a sensitivity analysis. Different values have been assigned to r_a using a trial and error procedure. An initial guess was made by trying small values for r_a, for example, for r_a = 0.15, thirty-four clusters were obtained, but produced a singular matrix, making it impossible to optimize the method using the Takagi-Sugeno procedure (Takagi and Sugeno 1985). As such, least-square optimization could not be carried out. Subsequently, the guess shifted towards the middle section of the normalized data and a sensitivity analysis was carried out in the 0.5 to 0.7 range, with an increment of 0.05. Errors during training and testing were recorded for each r_a as per Table 5.5. In the

developed model, r_a value was set to be 0.65 so as to minimize training and testing errors as shown in Figure 5.6-a. Accordingly, two clusters were obtained for the *Haul_Mid* scenario (see Figure 5.6-b).

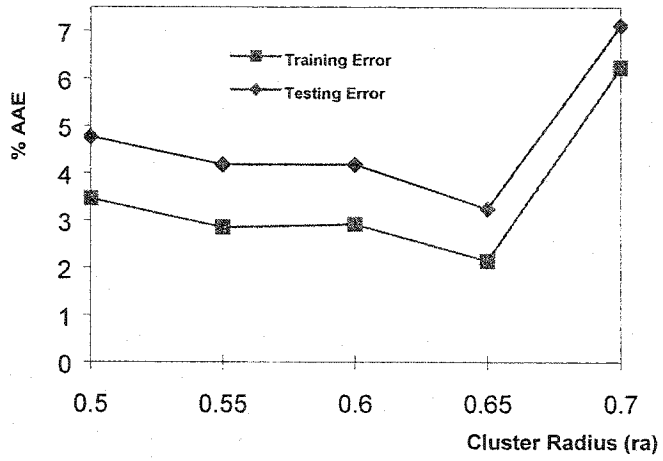


Figure 5.6-a Clusters' Radius versus Average Absolute Error

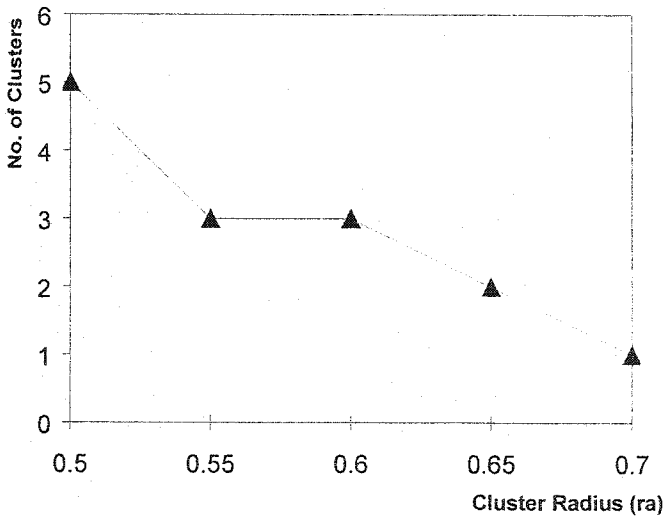


Figure 5.6-b Clusters' Radius versus their Number

Table 5.5 Results of Sensitivity Analysis

r_a	r_b	α	β	No. of Clusters	Data Point of Clusters					AAE of Training Data(%)	AAE of Test Data(%)
					1st	2nd	3rd	4th	5th		
0.5	0.75	16	7.111	5	118	67	140	86	123	4.77	3.46
0.55	0.825	13.223	5.877	3	118	140	123			4.18	2.86
0.6	0.9	11.111	4.938	3	118	123	140			4.18	2.92
0.65	0.975	9.467	4.208	2	118	152				3.25	2.15
0.7	1.05	8.163	3.628	1	28					7.12	6.24

The reduced potential neighborhood (r_b), threshold values for acceptance ($\bar{\varepsilon}$), and rejection ($\underline{\varepsilon}$) were set to be 0.975, 0.5 and 0.15, respectively. These values were adopted from Chiu (1994). It has been recommended to assign r_b a value 50% higher than r_a in order to avoid closely spacing cluster centers. For simplicity, the established values of r_a, r_b, $\bar{\varepsilon}$ and $\underline{\varepsilon}$ for the *Haul_Mid* scenario were used in the remaining seven scenarios.

Applying steps 1 and 2 of the algorithm, cluster centers were obtained for each scenario. Table 5.6 lists the number of clusters, their data points (from the training data sets) and the input vector (y*) for each cluster in the eight scenarios. Fuzzy rules were then developed for each cluster and their respective membership functions were generated using the following equation:

$$\mu_I = e^{-\alpha\left\|y - y_i^*\right\|^2} \qquad (5.11)$$

In which $\alpha = \frac{4}{(0.65)^2} = 9.467$

Table 5.6 Cluster Centers for the Different Scenarios

Senario	No. of Clusters	Data Point	y*
Haul_Single	3	55	(0.5413, 0.142)
		118	(0.4783, 0.8176)
		163	(0.0166, 0.1044)
Return_Single	3	173	(0.4945, 0.1927)
		88	(0.5596, 0.9267)
		107	(0.0597, 0.7793)
Haul_First	2	29	(0.5588, 0.133)
		45	(0.2228, 0.7569)
Return_First	2	90	(0.4989, 0.2060)
		91	(0.5197, 0.757)
Haul_Last	2	94	(0.4609, 0.1381)
		88	(0.7572, 0.7982)
Return_Last	2	124	(0.4981, 0.186)
		98	(0.357, 0.7583)
Haul_Mid	2	115	(0.1098, 0.4597, 0.1098)
		149	(0.1709, 0.5225, 0.7987)
Return_Mid	4	34	(0.1817, 0.3033, 0.1817)
		118	(0.4176, 0.6068, 0.7793)
		112	(0.7793, 0.4675, 0.1425)
		138	(0.0836, 0.1079, 0.7572)

The un-optimized (UO) output Z for the developed travel time is calculated using the following Equation:

$$Z = \frac{\sum_{i=1}^{c} \mu_i Z_i^*}{\sum_{i=1}^{c} \mu_i} \tag{5.12}$$

In which Z_i^* is the output vector for cluster i.

Takagi-Sugeno polynomial functions of 0th order (OTS_0) [$\boldsymbol{z}_i^* = \boldsymbol{h}_i$] and 1st order (OTS_1) [$\boldsymbol{z}_i^* = \boldsymbol{G}_i \boldsymbol{y} + \boldsymbol{h}_i$] were obtained by applying least-square optimization. The fuzzy inference system output (Chiu 1994) is obtained as follows:

$$Z = \sum_{i=1}^{c} \rho_i (G_i y + h_i) \tag{5.13}$$

In which ρ_i is calculated for each cluster as follows: $\rho_i = \frac{\mu_i}{\sum_{j=1}^{c} \mu_i}$

Table 5.7 lists the Z_i^* for UO, OTS_0 and OTS_1 models for the eight scenarios.

Table 5.7 Clusters' Output Vector

Cluster No.	Cluster No.	Z_i^*		
		UO	*OTS_0*	*OTS_1*
Haul_Single	1	14.32	13.64	(0.00028, 0.933059)y – 0.730892
	2	51.30	49.96	(0.00592, 0.72839)y – 8.27753
	3	11.00	14.20	(0.003528, 0.867688) – 0.86620
Return_Single	1	25.65	24.05	(0.00057, 0.89041)y + 0.36766
	2	57.34	60.34	(0.00371, 1.03205)y – 20.71207
	3	37.10	35.80	(0.02289, 0.40414) + 1.99675
Haul_First	1	14.36	13.13	(0.00035, 0.85441)y + 0.46132
	2	39.69	45.87	(0.00614, 0.34666)y+ 13.80752
Return_First	1	26.33	22.57	(-0.00012, 0.92725)y + 1.59780
	2	52.37	54.20	(0.00358, 0.76498)y – 1.1980
Haul_Last	1	14.30	13.95	(0.00011, 0.97507) – 0.09899
	2	51.64	51.05	(0.00466, 0.85816) – 12.97779
Return_Last	1	25.48	22.51	(0.00015, 0.80267)y + 3.53126
	2	51.33	53.22	(0.00390, 0.53547)y + 12.91802
Haul_Mid	1	12.57	12.56	(-0.00197, -0.00015, 0.98143)y + 0.49325
	2	53.19	51.83	(0.07763, 0.00286, 0.79518)y – 2.51499
Return_Mid	1	25.54	21.53	(0.02664, 0.00009, 0.87981)y + 1.57283
	2	55.54	59.28	(0.09888, 0.00113, 0.95916)y – 6.80637
	3	23.55	24.63	(0.00608, -0.00005, 0.99803)y – 0.26496
	4	48.00	50.94	(0.40132, 0.01068, 0.53514)y + 0.36865

5.2.7 Analysis of Results

Haulers' travel time values were estimated for the 36 randomly generated test cases. The travel time for each case was calculated using un-optimized fuzzy, OTS_0 and OTS_1 methods and compared to that calculated using the FPC software. Table 5.8 shows the sum of squares of errors (SSE), the standard error

(S_ε) and the average of the absolute errors (AAE) for each method used. The results indicate superiority of OTS_1 over the two methods, yielding least error in all scenarios considered. The implementation code for OTS_1 is included in Appendix A.

Table 5.8 Comparison of Errors in Estimating Travel Time

Model	SSE $\sum_{n=1}^{30}(t_{m_n} - t_{FPC_n})^2$	Sε $\sqrt{\frac{SSE}{n}}$	AAE $\frac{\sum_{n=1}^{30}(t_{m_n} - t_{FPC_n}) * 100 / t_{m_n}}{n}$
Haul_Single_UO	971.388	5.19	25.70
Haul_Single_OTS_0	894.42	4.98	24.49
Haul_Single_OTS_1	3.72	0.32	5.09
Return_Single_UO	59.89	1.29	15.67
Return_Single_OTS_0	47.88	1.15	14.76
Return_Single_OTS_1	0.81	0.15	3.59
Haul_First_UO	657.45	4.27	25.58
Haul_First_OTS_0	607.03	4.11	24.70
Haul_First_OTS_1	12.55	0.59	3.92
Return_First_UO	44.06	1.11	13.82
Return_First_OTS_0	22.60	0.79	11.53
Return_First_OTS_1	0.37	0.10	4.40
Haul_Last_UO	920.23	5.06	27.38
Haul_Last_OTS_0	872.82	4.92	27.57
Haul_Last_OTS_1	5.81	0.40	4.24
Return_Last_UO	48.80	1.16	11.43
Return_Last_OTS_0	22.49	0.79	9.25
Return_Last_OTS_1	0.90	0.16	3.98
Haul_Mid_UO	586.77	4.04	15.98
Haul_Mid_OTS_0	584.96	4.03	16.34
Haul_Mid_OTS_1	3.19	0.30	3.25
Return_Mid_UO	43.54	1.10	15.28
Return_Mid_OTS_0	22.36	0.79	12.25
Return_Mid_OTS_1	0.35	0.10	1.37

Figures 5.7 to 5.14 illustrate a comparison between FPC estimation and those generated using the Takagi-Sugeno 1st order models for the eight scenarios.

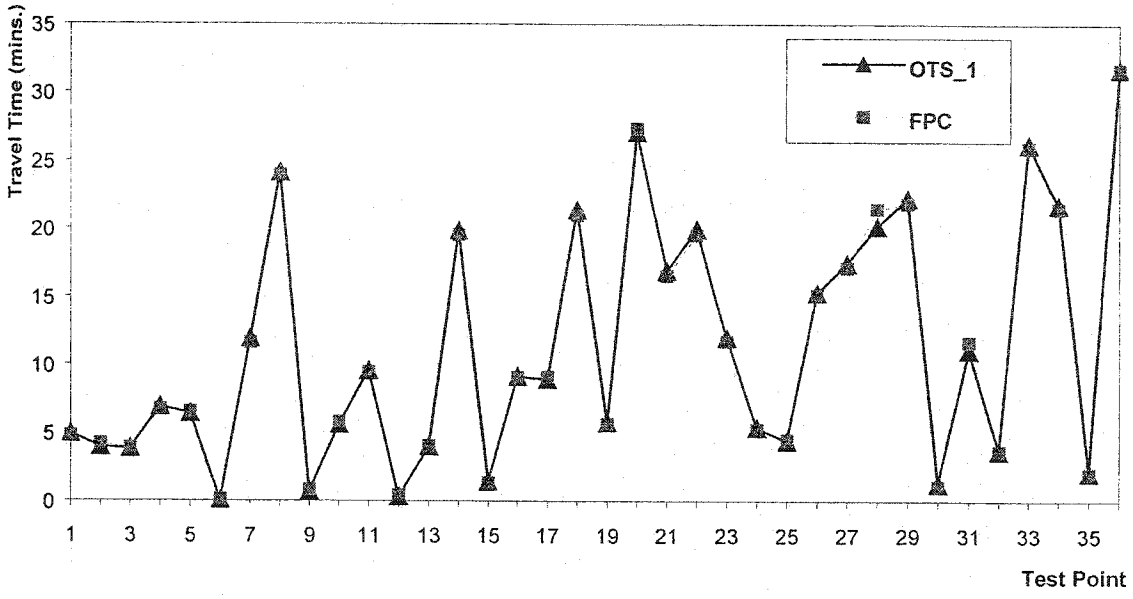


Figure 5.7 Comparison of Travel Time Generated Using FPC and Haul_Single

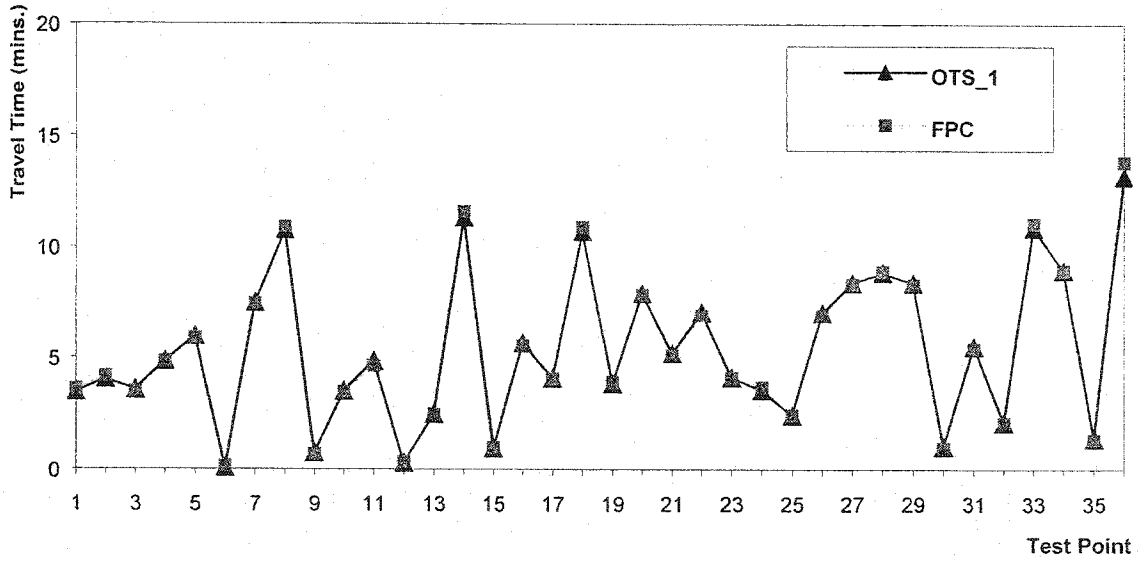


Figure 5.8 Comparison of Travel Time Generated Using FPC and Return_Single

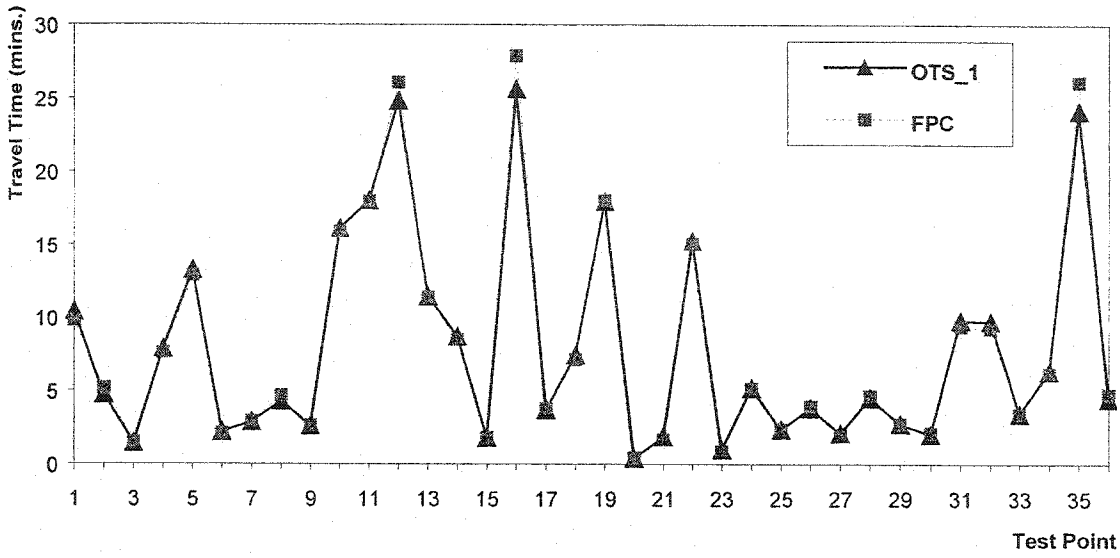


Figure 5.9 Comparison of Travel Time Generated Using FPC and Haul_First

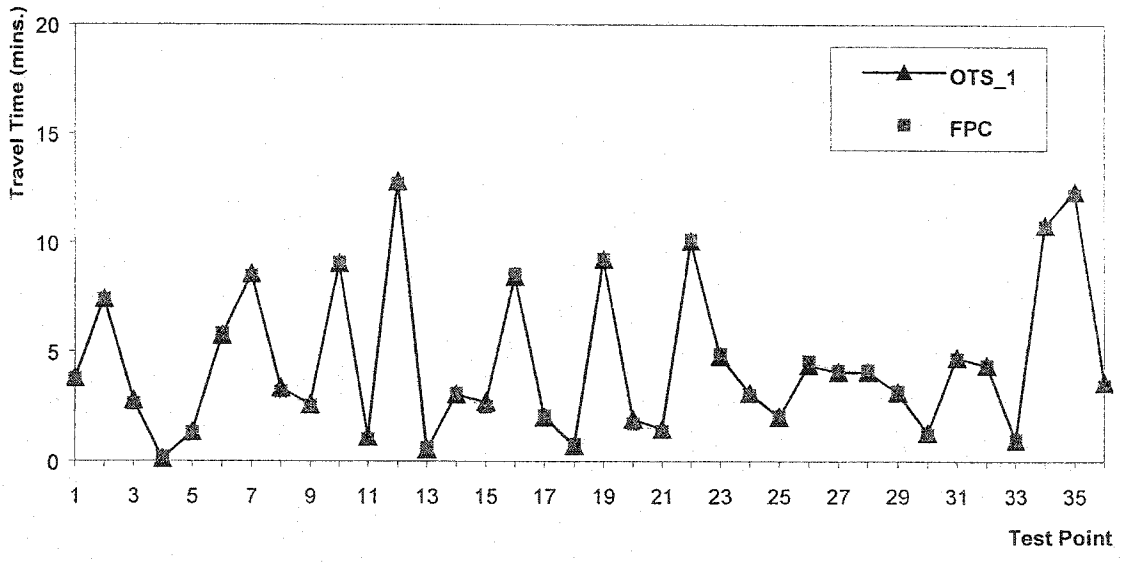


Figure 5.10 Comparison of Travel Time Generated Using FPC and Return_First

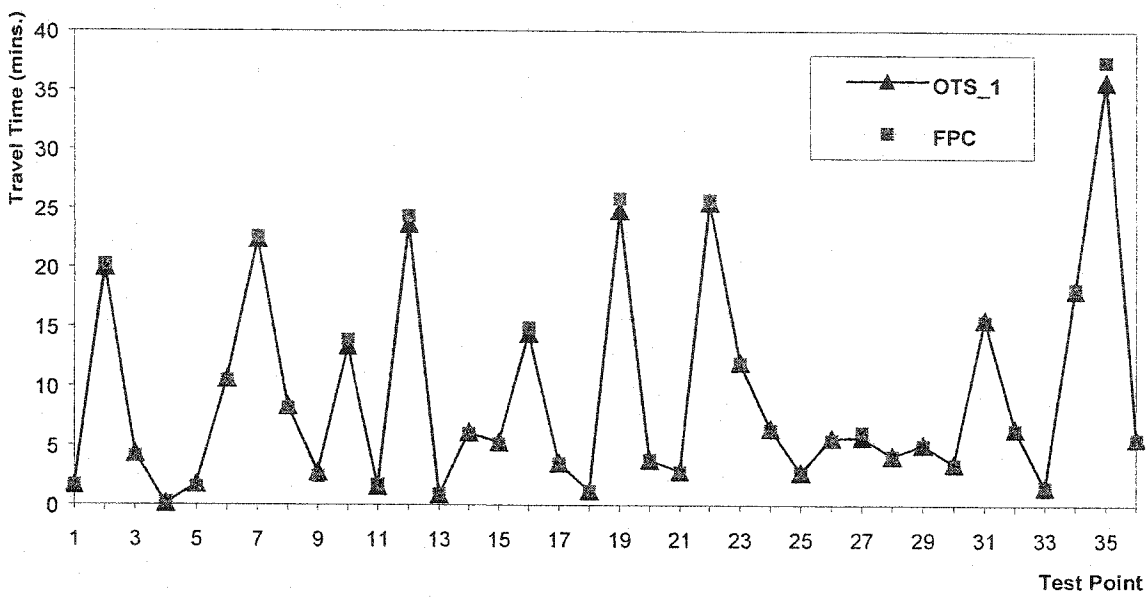


Figure 5.11 Comparison of Travel Time Generated Using FPC and Haul_Last

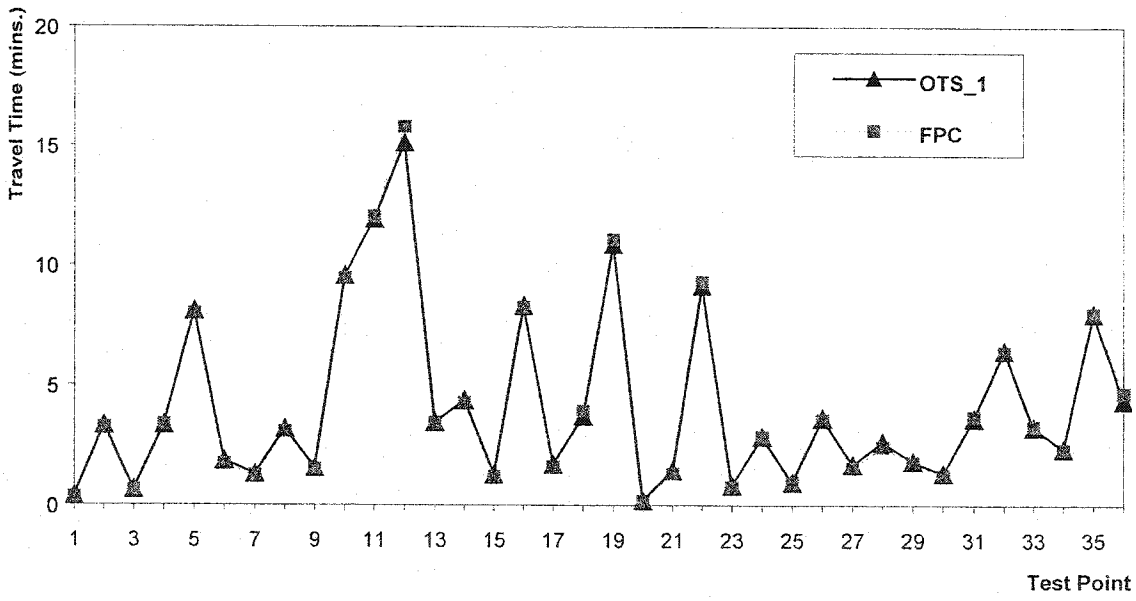


Figure 5.12 Comparison of Travel Time Generated Using FPC and Return_Last

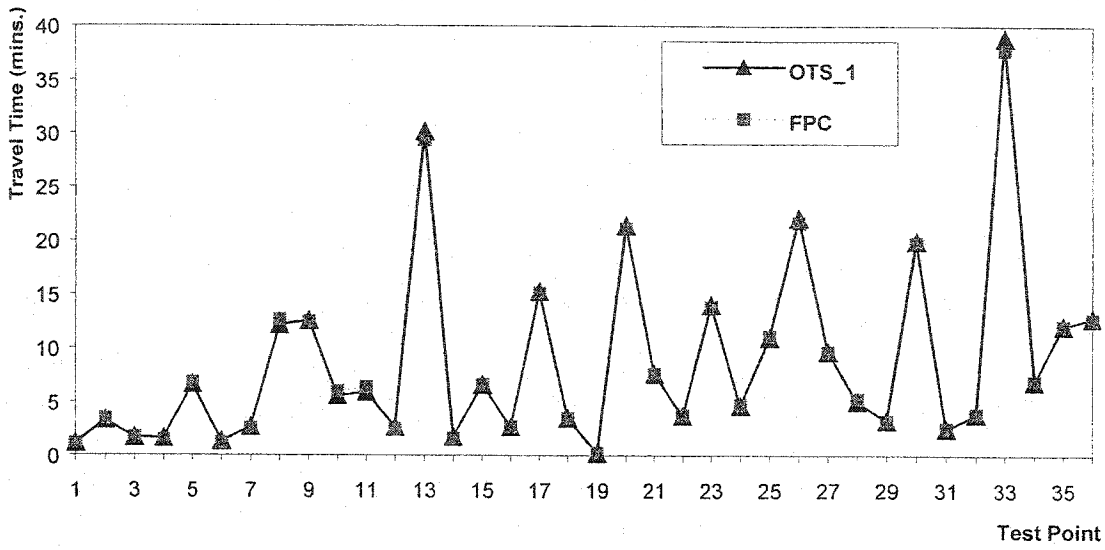


Figure 5.13 Comparison of Travel Time Generated Using FPC and Haul_Mid

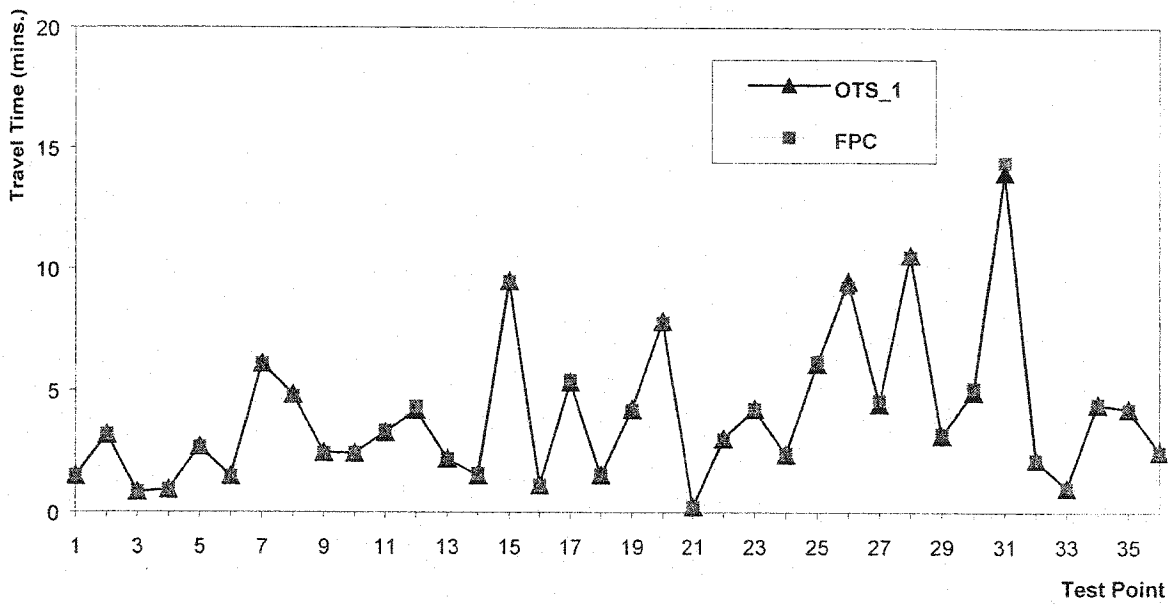


Figure 5.14 Comparison of Travel Time Generated Using FPC and Return_Mid

5.2.8 Numerical Example

It is required to estimate the travel time of an off-highway truck hauling earth across an intermediate road segment under the following conditions: 1) previous segment allowable speed = 28.71 km/hr, 2) segment length = 3,473 m, 3) segment allowable speed = 67.06 km/hr, and 4) segment total resistance = 0%. In this example, the method developed in this chapter is applied for estimating the required travel time and the results are compared to those generated utilizing FPC.

The data given for the previous segment allowable speed, segment length and segment allowable speed were first normalized as 0.37336, 0.69429 and 1, respectively. Using the model developed for the *Haul_Mid* scenario which consists of two clusters, their respective membership functions can be expressed using Equation (5.11) as:

$$\mu_1 = e^{-9.467\left\| y - y_1^* \right\|^2} \tag{5.14}$$

$$\mu_2 = e^{-9.467\left\| y - y_2^* \right\|^2} \tag{5.15}$$

Substituting the values for y (0.37336, 0.69429, 1), y_1^* (0.1098, 0.4597, 0.1098) and y_2^* (0.1709, 0.5225, 0.7987) in Equations 5.14 and 5.15, the membership

values were calculated to be 0.00017 and 0.34957 for μ_1 and μ_2, respectively. The estimated average speeds for the two clusters were calculated as follows:

$$z_1^* = (28.71,3473,67.06) \bullet (-0.00197,-0.00015,0.98143) + 0.49325 = 65.73$$

$$z_2^* = (28.71,3473,67.06) \bullet (0.07763,0.00286,0.79518) - 2.51499 = 62.97$$

Using the values obtained above (μ_1, μ_2, z_1^* and z_2^*) and Equation (5.12), the average speed was calculated to be 62.96 Km/hr, and accordingly, the travel time to be 3.31 mins.

The hauler's travel time was also estimated, using FPC, to be 3.44 mins. The average speed, for the case under study, is calculated using Simphony (Equation 2.11, Simphony 2000), ignoring hauler's acceleration from 28.71 to 67.06 km/hr, as follows:

Speed = 67.06*[(55-0)/55]=67.06 Km/hr

Accordingly the travel time is estimated to be 3.11 mins. Table 5.9 shows the absolute error, compared to FPC, for the estimated travel time using the proposed method and that used in Simphony.

Table 5.9 Comparison of Errors in HTTA and Simphony

Model	Travel Time (mins.)	AE ($\frac{\lvert t_{Model} - t_{FPC} \rvert}{t_{FPC}}$ *100)
HTTA	*3.31*	*3.78*
Simphony	*3.11*	*9.59*

5.3 Earthmoving Markup Application

5.3.1 Background

Contractors frequently face two main questions in the bidding process: 1) whether to bid or not, and 2) what markup to allocate, if decided to bid. The first question can be addressed based on project characteristics, market conditions, and current workload of contractors. Wanous et al (2000), for example, developed a parametric model to assist in making that decision. This section focuses on the second question. A number of models have been developed, since the mid-1950s, for estimating markup. These could be grouped in: 1) statistical models; 2) multi-attribute utility theory (MAUT) and analytic hierarchy process (AHP) type models; and 3) artificial intelligence (AI) models. The models of the first group (e.g. Friedman 1956, Gate 1967, Carr 1982) lack the ability to capture actual bid characteristics including, but not limited to, project complexity, project duration and market conditions.

The models of the second group (e.g. Ahmed and Minkarah 1987, Seydel and Olson 1990, Dozzi et al 1996, Chua and Li 2000, Cheung et al 2001) restrict the user to a rigid, predefined, decision hierarchy. Further, they may either require the use of commercial software as in the case of Chua and Li (2000) or not account for the relative importance among attributes as in the case of Ahmed and Minkarah (1987). The models of the third group utilize knowledge-base expert systems (KBES) (Ahmed and Minkarah 1988-a), and neural networks

(NNs) (Moselhi and Hagazy 1993-b). In the case of KBES, it is difficult to extract the knowledge in this domain and express it in the form of IF....THEN type rules. NNs, although powerful in this domain, require retraining with evolving market conditions and adaptation to suit the bidding strategy of each contractor.

5.3.2 Proposed EMMD

The following sections present a model for estimating earthmoving markup percent. The earthmoving markup application (*EMMA*) combines the advantages of both multi-attribute utility theory "MAUT" (Keeney and Raiffa 1993, Hatush and Skitmore 1998) and the analytic hierarchy process "AHP" (Saaty 1982). *EMMA* is generic, designed for use as a stand-alone application or as an integrated application within *SimEarth*. It can also be used as a tool for evaluating bid proposals. *EMMA*, unlike those referred to above, provides users with the flexibility to construct the decision hierarchy in such a way so as to suit each contractor's business environment. The model also: 1) provides the user with the flexibility of selecting utility functions for all involved attributes, in a way that captures the decision-maker's attitude towards risk (averse, neutral or prone); and 2) can be used to evaluate competing alternatives, including bid proposals.

EMMA is dedicated for estimating markup percent including profit and contingency margins. It estimates markups, accounting for the risk attitudes of decision-makers. Alternatively, the user can specify, without using *EMMA,* the markup that suits his business plans and enter it directly to *SimEarth* as depicted

in Figure 5.15. The decision criteria and its associated utility functions are developed using multi-attribute utility theory. The relative weights associated with the attributes considered in the decision criteria are established using the analytic hierarchy process. Figure 5.16 provides a schematic diagram, outlining the input and output variables of *EMMA*. Detailed description of developments made in *EMMA* can be found in Appendix C.

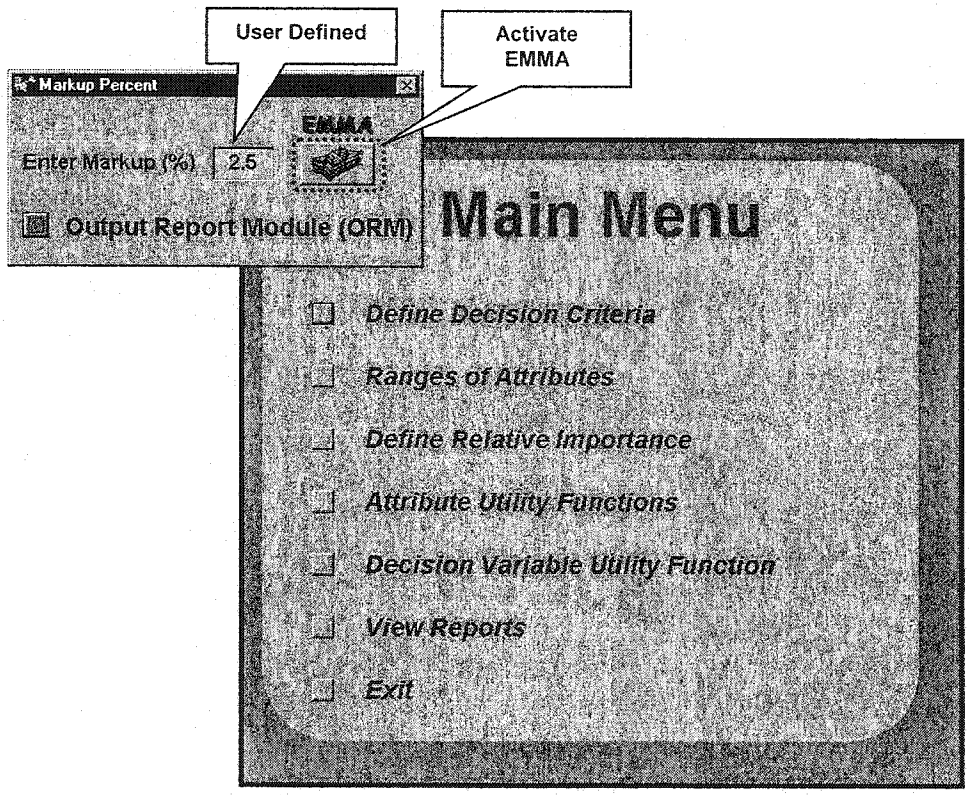


Figure 5.15 EMMA Main User Interface

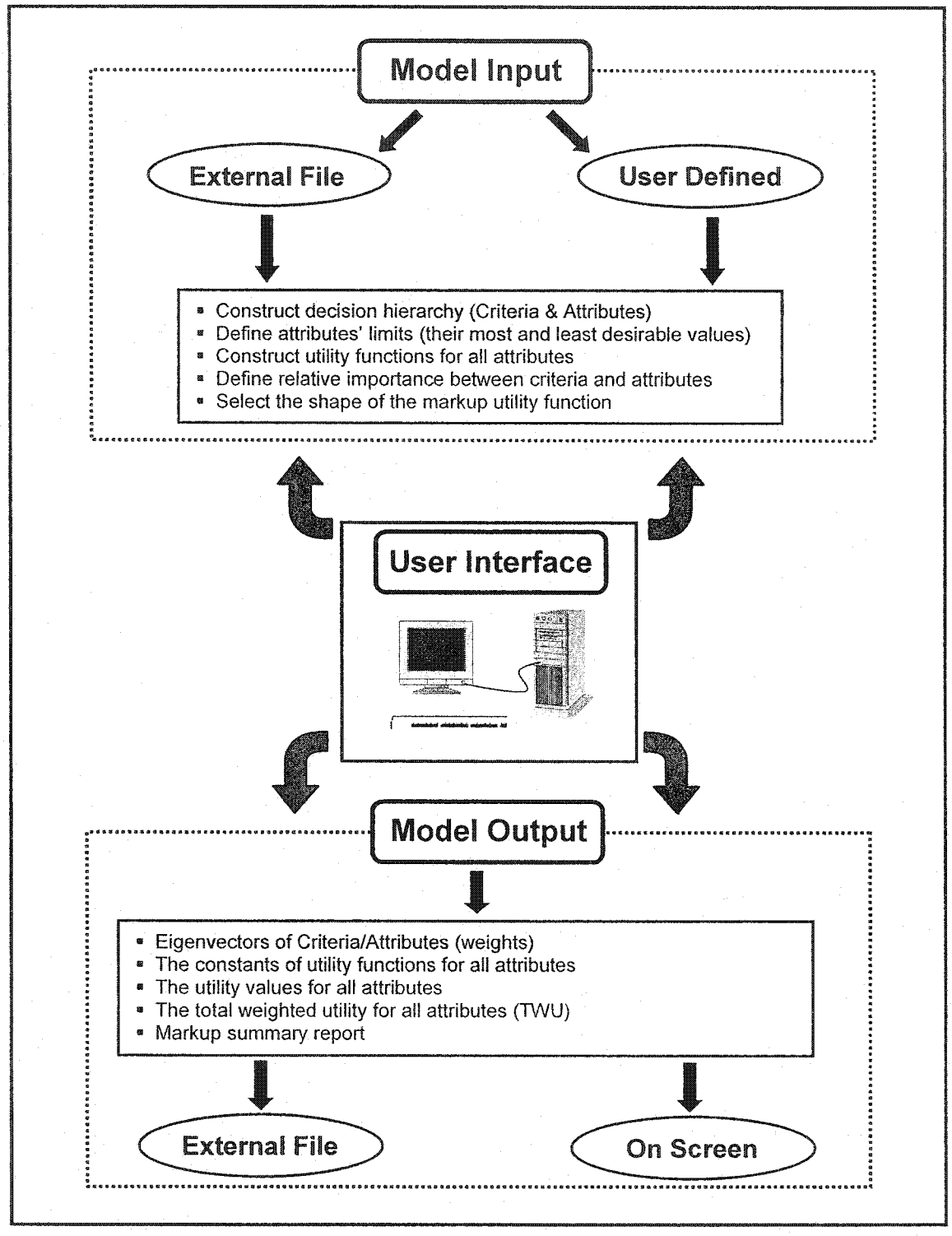


Figure 5.16 Model Input/Output Diagram

5.4 Summary

This chapter presented an application (*HTTA*) for estimating haulers' travel time which is integrated as an auxiliary with the developed *SimEarth*. Factors affecting such estimation have been identified and analyzed utilizing linear regression and 180 randomly generated data cases. *HTTA* accounts for the different types of road segments and for acceleration and deceleration in transition zones. Un-optimized (UO) subtractive clustering, optimized Takagi-Sugeno 0^{th} (OTS_0) order subtractive clustering and optimized Takagi-Sugeno 1^{st} order subtractive clustering were tested and evaluated for possible use in the developed method. The optimized Takagi-Sugeno 1^{st} (OTS_1) order subtractive clustering was found to outperform the others for all types of segments. To validate the developed *HTTA*, thirty-six randomly generated data cases were analyzed and compared to those generated using FPC. The results were found in close agreement, indicating good accuracy of the proposed application. A numerical example was presented in order to demonstrate the use of the proposed method.

The chapter also presented an application (*EMMA*) for estimating markup. *EMMA* was developed utilizing the analytic hierarchy process and multi attribute utility theory. The developed application provides its users with flexibility in constructing the decision hierarchy that suits their strategies. It also accounts for the different risk attitudes (averse, neutral and averse). The model has been coded utilizing MS visual basic 6.0.

CHAPTER 6

EARTHMOVING GENETIC ALGORITHM

6.1 General

This chapter presents a methodology for simulation optimization utilizing genetic algorithms. The methodology is applied to a newly developed simulation-based system for estimating time and cost of earthmoving operations. A genetic algorithm was developed to search for a near-optimum fleet configuration that reduces project total cost. The algorithm considers a set of qualitative and quantitative variables that influence the production of earthmoving operations. Qualitative variables represent the models of equipment used in each fleet scenario, whereas quantitative variables represent the number of equipment involved in each scenario. Pilot simulation runs are carried out for all configurations generated by the developed algorithm. Complete simulation analysis is then performed for the fleet recommended by the algorithm.

The developed algorithm accounts for: 1) indirect cost and 2) qualitative and quantitative variables involved in these operations. The developed genetic algorithm has a powerful computational utility that speeds up the calculations. This is accomplished by employing the elitism function and by storing the fitnesses of chromosomes in a built-up database to avoid duplication of fitness

calculation, should those chromosomes appear in future generations. In addition, the algorithm accounts for the consideration of practical fleet configurations that incorporate the judgment and experience of contractors. The algorithm has been implemented using MS Visual C++ 6.0. A numerical example is presented to demonstrate the different features of the developed algorithm and to illustrate its capabilities in selecting near-optimum fleet configurations.

6.2 Proposed Algorithm

6.2.1 General Overview

The proposed algorithm utilizes genetic algorithms (Holland 1992, Coley 1999) and assists in selecting near-optimum fleet configurations that minimize the total cost of those operations. It is integrated with two components of *SimEarth*: 1) the EarthMoving Simulation Program (*EMSP*) to calculate the fitness of the created chromosomes as will be described later, and 2) the Output Reporting Module (*ORM*) to present the optimization results. Figure 6.1 depicts the components of the proposed algorithm and data flow through its components. The following section provides an overview of the developed algorithm.

6.2.2 Fleet Representation

Based on the task at hand, a set of fleet scenarios are defined, each

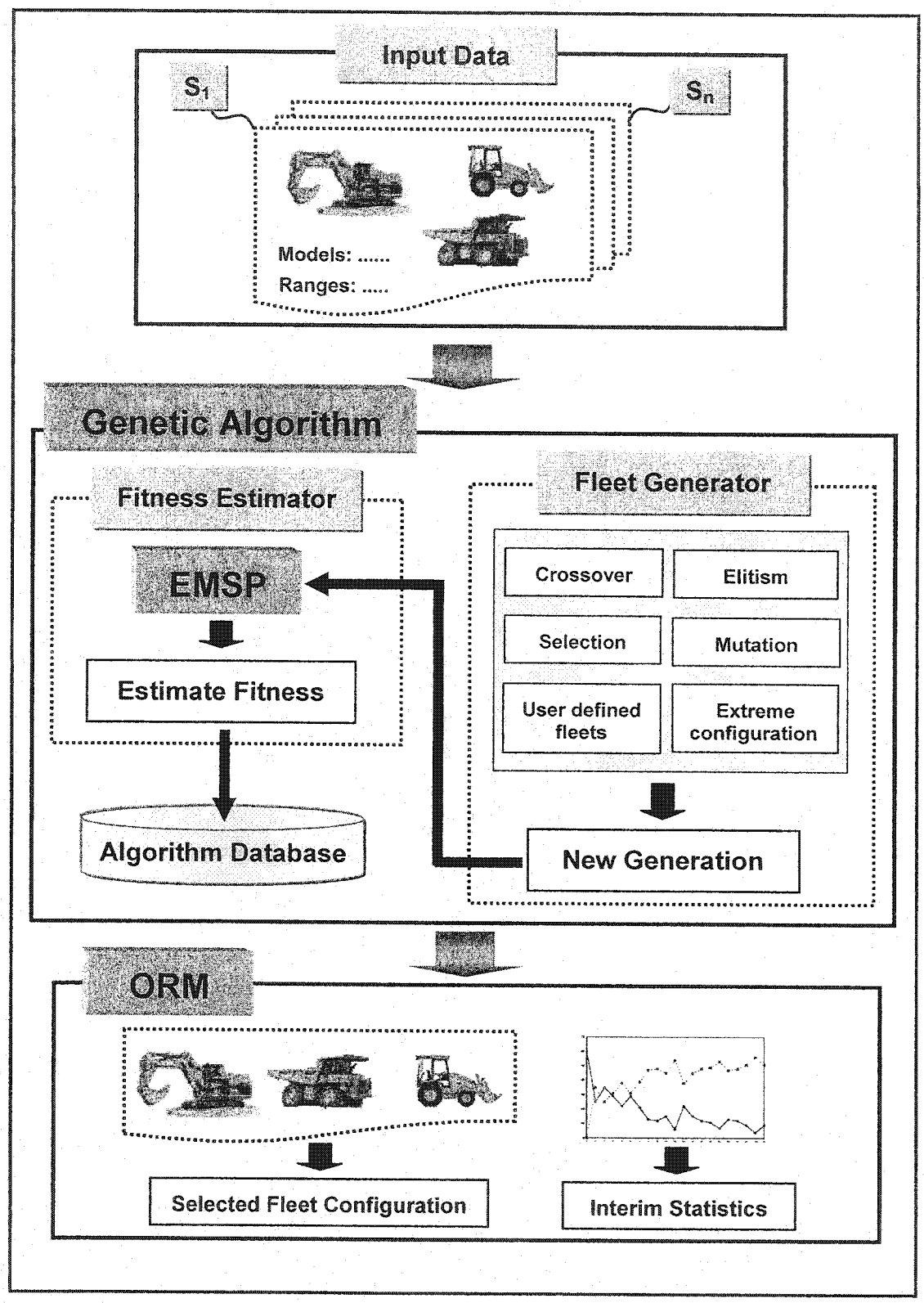


Figure 6.1 Components of the Proposed Algorithm

representing the type of equipment used along with their respective models. Fleet configurations, in the developed algorithm, are generated from the defined fleet scenarios by articulating the number of equipment involved. Each fleet scenario and the equipment models included in that fleet are considered qualitative variables. The number of equipment in each configured fleet scenario is considered quantitative variable. Each chromosome in the developed algorithm represents a fleet configuration as shown in Figure 6.2. The entire set of chromosomes considered is referred to here as a population. The population is grouped into a set of sub-populations, each representing a number of fleet configurations that belongs to a specific scenario. Each chromosome consists of a number of genes (see Figure 6.2). The first gene in each chromosome is reserved for an index, identifying the sub-population it belongs to. The rest of the genes are dedicated for representing the number of each type of equipment used in that fleet scenario. For example, for a fleet that consists of haulers, loaders, spreaders and compactors, the chromosome representing this fleet has five genes. The first gene indicates the scenario number or index, and the last four chromosomes represent the number of haulers, loaders, spreaders and compactors, respectively. It should be noted that the model type of equipment and their respective characteristics are captured in the defined fleet scenario, i.e. in the sub-population that represents that scenario.

6.2.3 Computation Procedure

A flow chart that depicts the computational procedure of the developed algorithm is shown in Figure 6.3. Upon defining the input data of the algorithm, the first population is constructed by randomly creating a set of chromosomes, each representing a fleet configuration within the specified range of the number of equipment used. Predefined chromosomes can also be considered in the first population to ensure the inclusion of practical fleet configurations. In addition, upper and lower number of equipment can also be used to configure extreme fleets since a near-optimum fleet may exit at these boundaries.

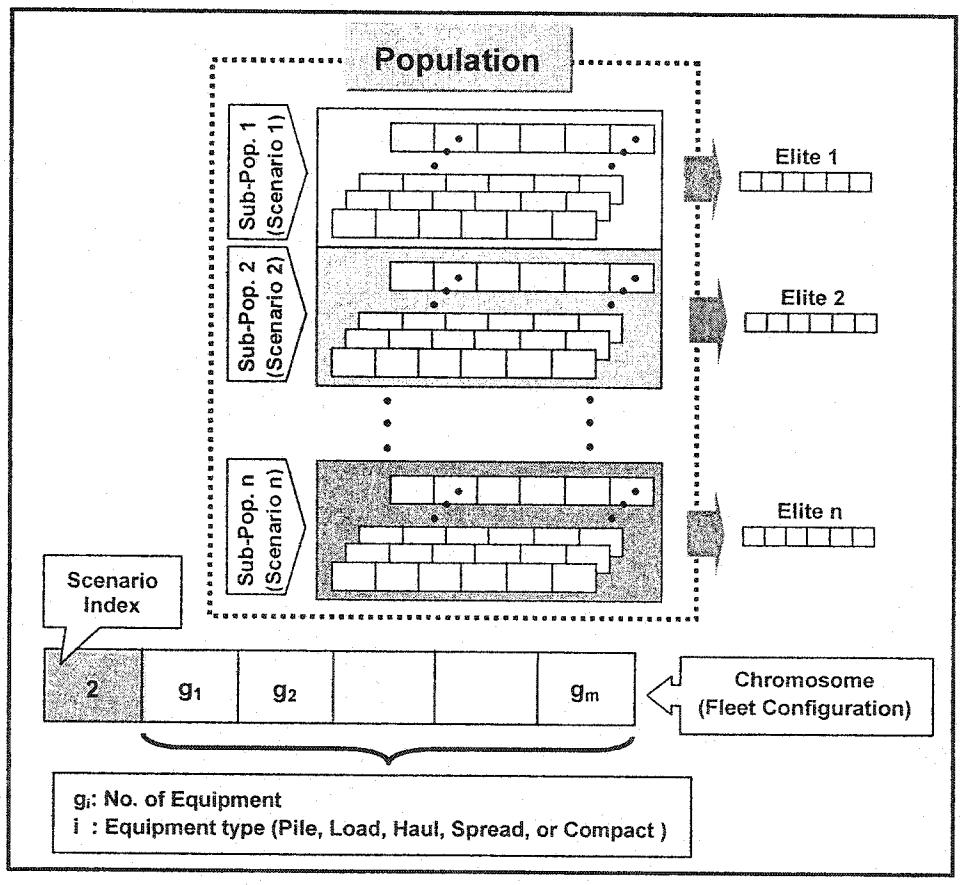


Figure 6.2 Population and Chromosome Representation

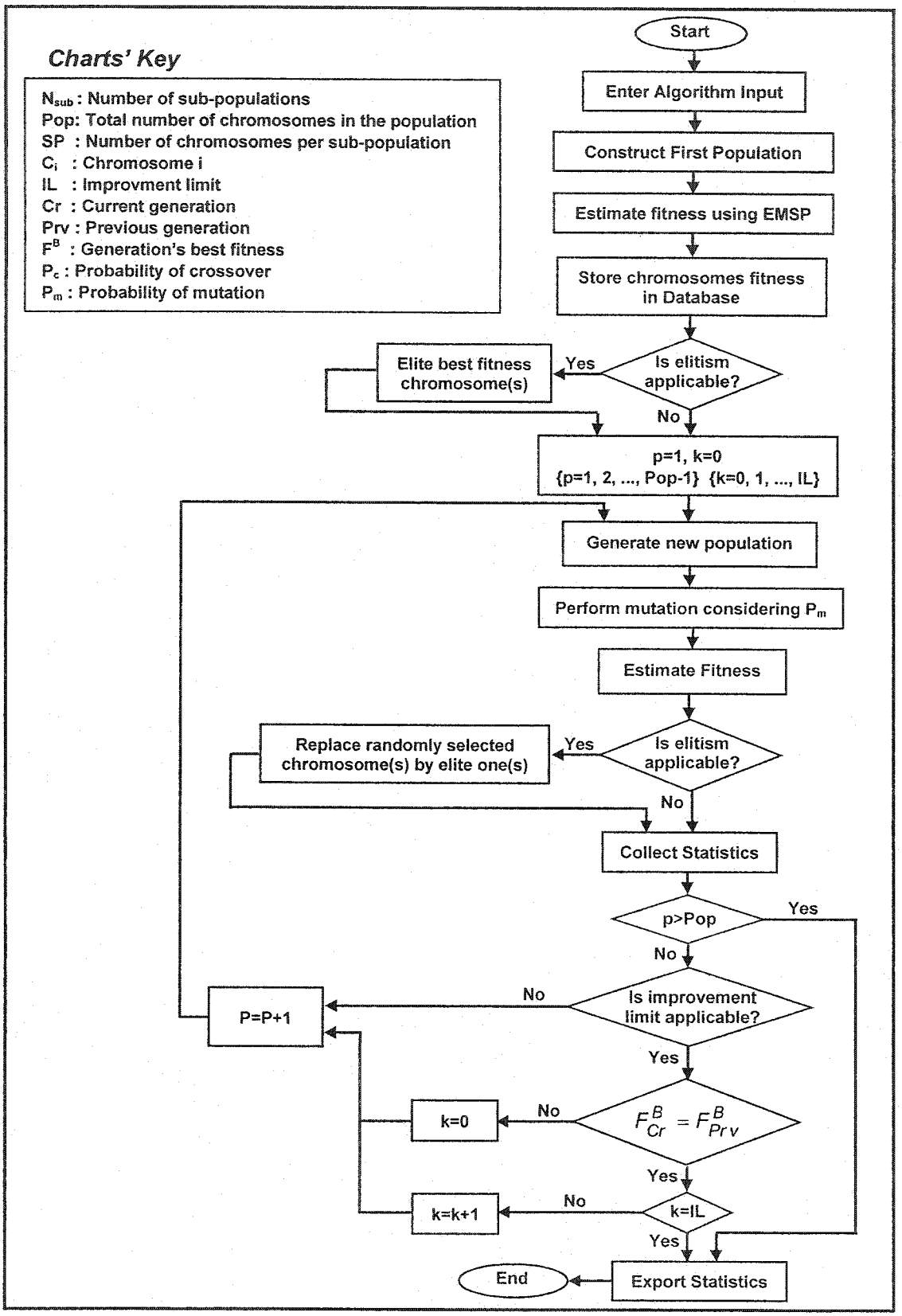


Figure 6.3 Flowchart of the Computational Process

Upon completion of generating the chromosomes of the first population, the fitness of each chromosome is calculated by *EMSP*. Specifically, *EMSP* is used to perform simulation in order to estimate the daily production *(D_Prod)* and, accordingly, project duration which directly impact the project total cost as shown in Equation 6.1. It should be noted that the project total cost represents the fitness of chromosomes as follows:

$$F=\{DH*[\sum_{i=1}^{t} EHC_i * NE_i *(TQ/D\_Prod)]\}+\{[TQ/(D\_Prod*MD)]*TR\_IC + TI\_IC\} \quad (6.1)$$

In which;

F : Estimated Fitness ($)

DH : Scheduled hours per day (hrs.)

t : number of types of equipment used

EHC : Equipment hourly cost ($/hr.)

NE : Number of equipment associated with each type

TQ : Total quantity of earth to move (m^3)

MD : Scheduled days per month (days/month)

TR_IC : Time-related indirect cost ($/month)

TI_IC : Time-independent indirect cost ($)

The first term of Equation 6.1 represents the direct cost and the second term represents the indirect cost. The indirect cost has two components; time related

and time independent. The algorithm provides the user with the flexibility in defining the two components, accounting for the project characteristics. After estimating the fitness for all chromosomes in the first populations, the chromosomes and their fitness are stored in the database of the algorithm to avoid performing simulation analysis for chromosomes which have been evaluated before, should they appear in future generations.

In each generation, the chromosomes which have best fitness (least total cost) within each sub-population are identified and stored, if elitism is applicable (see Figure 6.2). Should these chromosomes have higher fitness values than those generated in the new sub-populations, they will automatically replace a randomly selected equivalent number of chromosomes in those sub-populations. While the algorithm searches for a solution, three main processes are carried out (see Figure 6.3). These are: 1) creating new generations, 2) performing mutation, and 3) estimating fitness values. The computation process is terminated when the algorithm reaches the maximum specified number of generations or if non-improvement in the fitnesses was achieved over a specified number of generations.

6.2.4 Generating of New Populations

In this process, offspring chromosomes are selected in pairs and moved to the new generation for each individual sub-population. This process depends on the number of sub-populations (N_{sub}) and the number of chromosomes (SP) in each

sub-population. It also depends on the specified threshold value for the crossover process (P_c). The flowchart shown in Figure 6.4 depicts this process. The chromosomes used in the process of crossover are selected according to the "*Roulette Wheel*" selection procedure (Goldberg 1989, Holland 1992, Coley 1999) which accounts for the relative fitness of each chromosome within each sub-population.

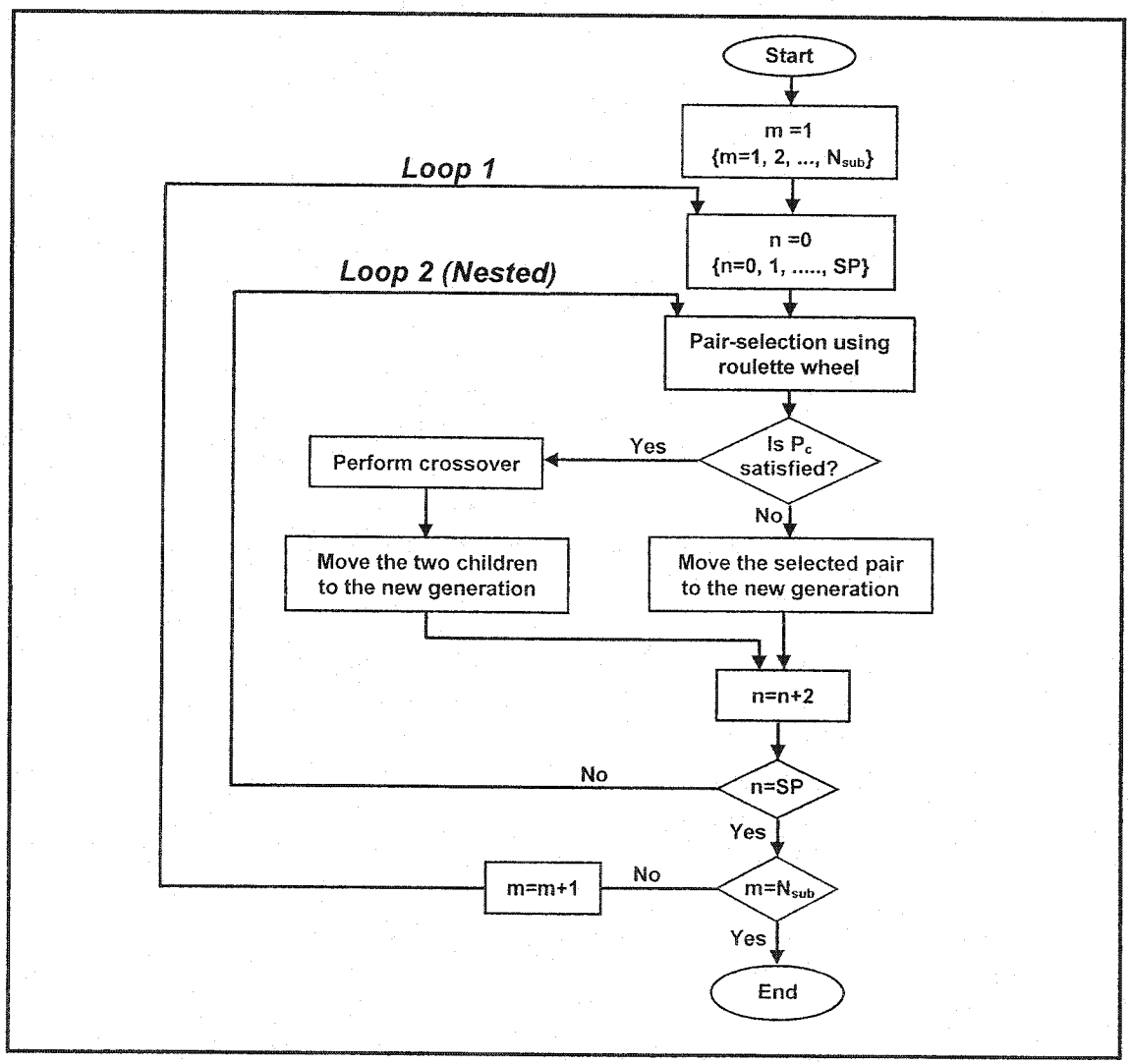


Figure 6.4 Flowchart for Generating New Populations

Since the "*Roulette Wheel*" procedure, referred to earlier, is biased towards the biggest fitness, a modification is carried out in the chromosomes' fitnesses in order to bias the selection towards chromosomes that have small fitness, thus, accounting for minimization rather than maximization. The modified fitness $[F^m(C_i)]$ is set equal to $[1/F(C_i)]$ in the proposed algorithm. Subsequently, the modified fitnesses are normalized $[F_N^m(C_i)]$ in such way that the summation of all normalized fitnesses, within a sub-population $[\Sigma F_N^m(C_i)]$, is equal to 1.0. According to the "*Roulette Wheel*" procedure, a random number is generated in the interval [0,1] as shown in Figure 6.5. By adding the normalized fitnesses of chromosomes one at a time, the chromosome selected is the one that when added, makes the summation greater than the generated random number. It should be noted that the normalized fitness $[F_N^m(C_i)]$ is used only for the purpose of selection and that the fitness $[F(C_i)]$ is utilized for all computations in the algorithm.

After selecting a pair of chromosomes, a random number is generated in the interval [0,1] and compared to P_c, if it is less than P_c: crossover is carried out; otherwise the two selected chromosomes are moved to the new generation without any change in their genes. The crossover process is the fundamental mechanism of GAs that makes them imitate biological genes (Holland 1992). It provides a means for exploring new alternatives in the solution space (Coley 1999). In the proposed algorithm, the crossover position is selected randomly after the second gene within a chromosome, as shown in Figure 6.6. The

"*Roulette Wheel*" selection process of the chromosomes is continued until the generation of the first sub-population is completed. The process is repeated for the rest of the sub-populations (Loop 1 in Figure 6.4).

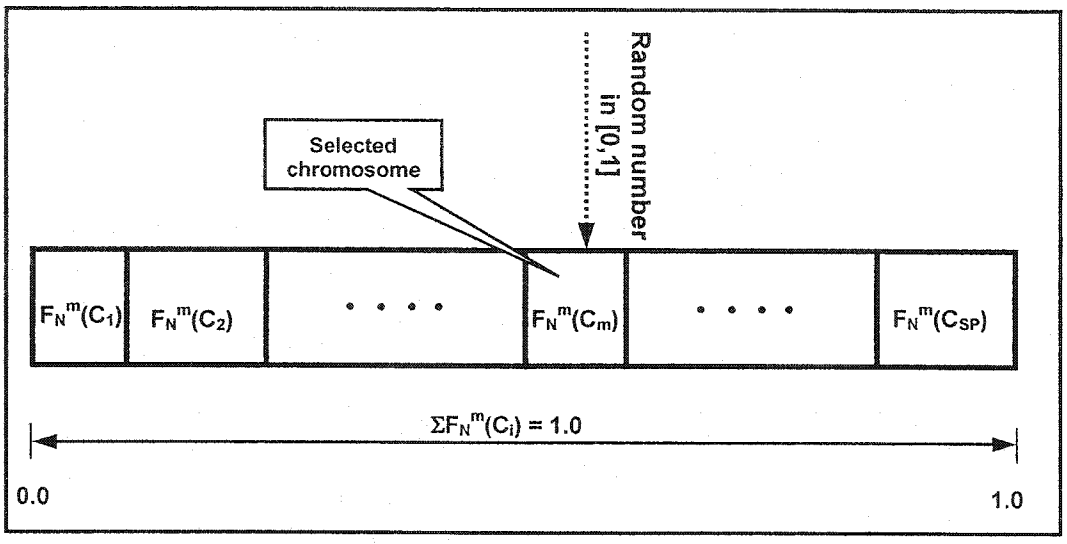


Figure 6.5 Random Selection of Chromosomes

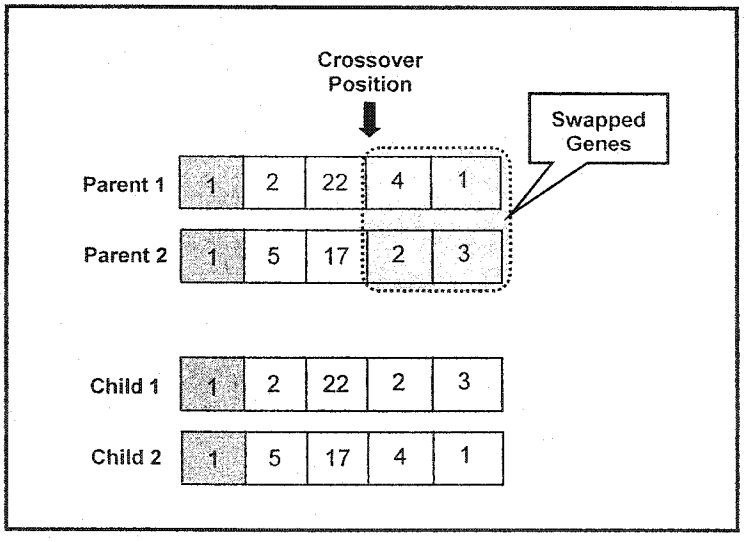


Figure 6.6 Crossover Process

6.2.5 Mutation Process

After the creation of the new population, the mutation process is carried out (see Figure 6.3) in an effort to avoid local minima and to ensure that newly generated populations are not uniform and incapable of further evolution (Holland 1992). In this process, a random number is generated in the interval [0,1] and compared to a specified threshold value (P_m): if it is less than P_m, mutation is carried out for that gene; otherwise the gene is skipped. Mutation is performed on all genes in each chromosome except the first. This is achieved by selecting a random integer number, within the specified range for that gene (representing the number of equipment used), and replacing the gene value by that number (see Figure 6.7).

6.2.6 Performance of the Algorithm

An effort has been made to improve the performance of the developed algorithm; enhance its conversion and reduced the required computation time. The fitness of each chromosome is stored to avoid repeating its computation at a later time, should that chromosome appears in future generations (see Figure 6.8). This has been introduced to speed up the computations of the algorithm by avoiding unnecessary simulation analysis to estimate project duration, necessary for computing the fitness of each chromosome. Elitism is employed to detect and retrieve top performing chromosomes and passing them to future generations. This is carried out for each fleet scenario (see Figure 6.2). Mutation is employed

to enhance convergence; avoiding local minima as described earlier. Of interest here to note that the algorithm tracks its performance and its rate of convergence through the "Non-Improvement Limit" parameter as shown in Figure 6.9.

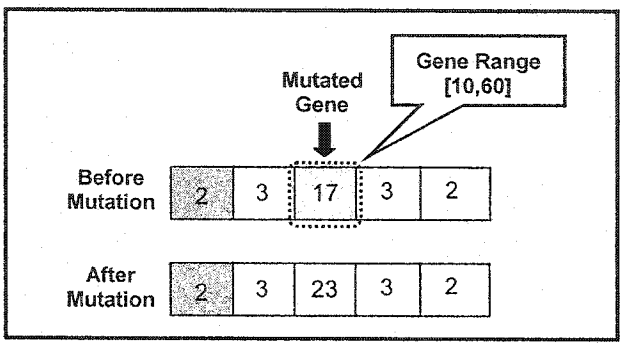


Figure 6.7 Mutation Process

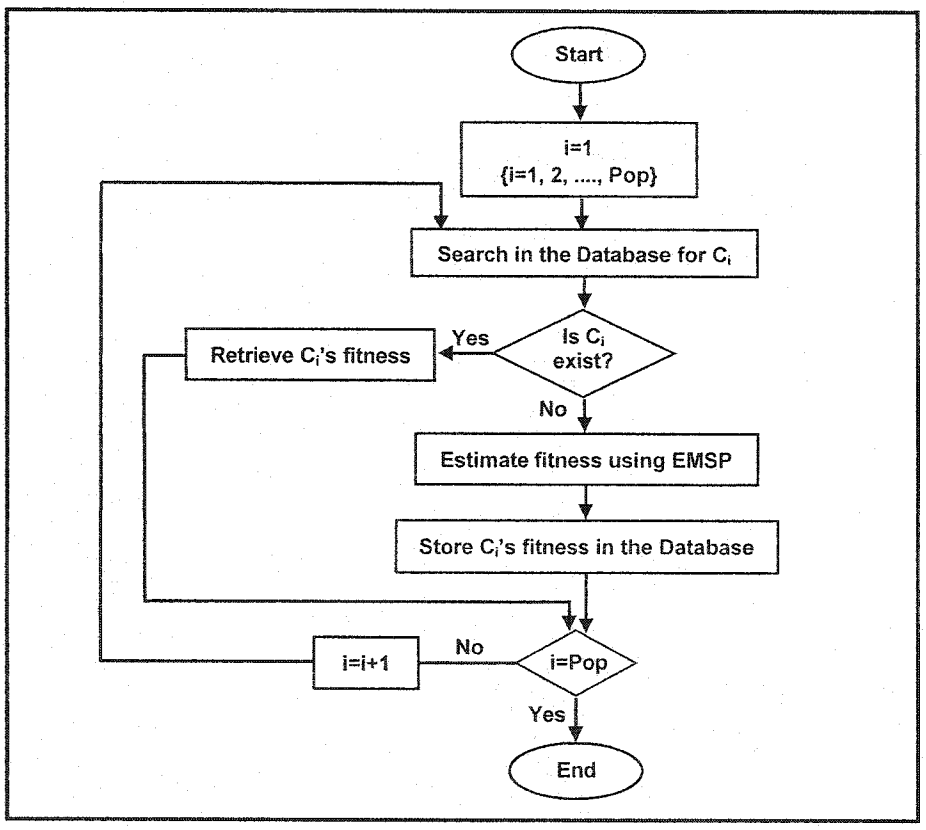


Figure 6.8 Flowchart for Fitness Estimating

6.2.7 Interim Statistics

Throughout the computational process, interim statistics related to the fitness of chromosomes are generated and stored. These statistics include: 1) the number of calculated and retrieved fitnesses, 2) the number of crossover and mutation processes and 3) the best and average fitnesses for all sub-populations in each generated population. These statistics are gathered and exported to the output reporting module (*ORM*).

6.3 Computer Assisted Implementation (EM_GA)

The algorithm described above, named *EM_GA*, has been coded using MS visual C++ 6.0 and runs within the *SimEarth* system. A set of interactive screens was designed using Microsoft visual basic 6.0 to facilitate data entry (see Figure 6.9). To proceed with the application of the algorithm, the user needs to specify the following: 1) the size of sub-population, 2) the threshold probability values of crossover and mutation, 3) whether the extreme fleet configurations (upper and lower number of equipment) are allowed or not, 4) whether the user is allowed to define his/her own fleet configuration(s), 5) whether elitism is allowed and 6) the termination criteria (see Figure 6.9). Extreme fleet configurations are allowed since solutions might exist at the boundary points of the solution space (Bosel et al 1999). On the other hand, a user-defined option is provided to allow experienced users to specify fleet configuration(s) that are practical and close to least cost conditions.

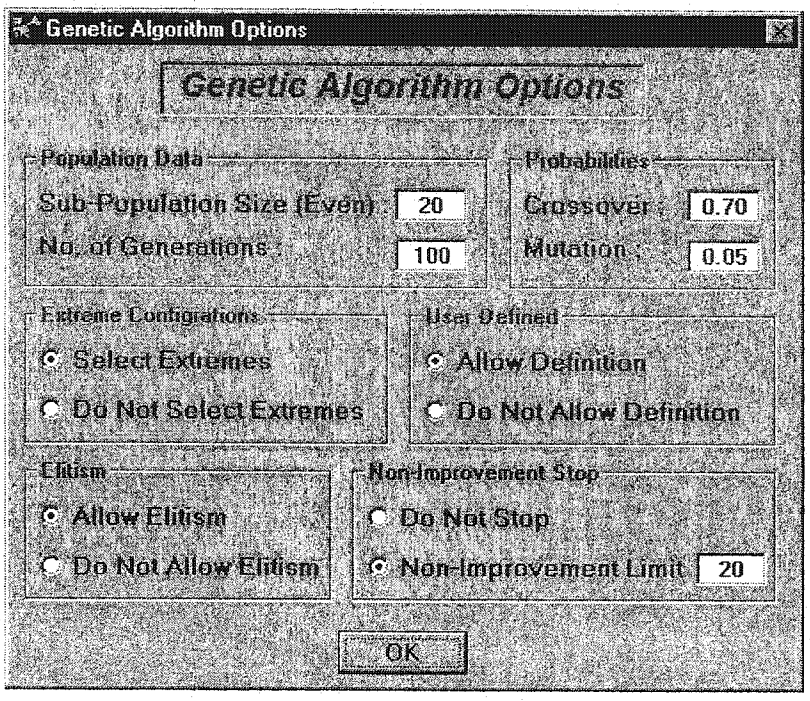


Figure 6.9 EM_GA Main User Interface

EM_GA receives its input data from the *SimEarth* environment into two protocols as outlined in Table 6.1. It should be noted that data pertaining to the parameters of the algorithm and the fitness function are passed directly to the main function of the algorithm, whereas the project and equipment data are imported by the algorithm from external files. The computer code of the developed *EM_GA* is included in Appendix A.

Figure 6.10 depicts the dialog box for the main menu of *ORM*, dedicated to reporting *EM_GA* results in different formats. *ORM* provides a number of text and graphical formats. The text reports simply list the fitnesses of chromosomes in the different generations along with the parameters used in the algorithm and their respective values (see Appendix B).

Table 6.1 Data Transformation to EM_GA

Passing Protocol	Type of Data
Passed to the main function	▪ Involved activities in current operation ▪ Sub-population size ▪ No. of fleet scenarios ▪ No. of generations ▪ Non-improvement limit ▪ Crossover probability ▪ Mutation probability ▪ Existence of second model of hauler ▪ Is elitism applicable ▪ Is extreme fleet configurations applicable ▪ Is user-defined fleet configurations applicable ▪ Scheduled daily hours ▪ No. of working days per month ▪ Time-related indirect cost ($/month) ▪ Time-independent indirect cost($)
External files	▪ Ranges of equipment for all involved fleet scenarios ▪ Hourly owing and operating cost ▪ Scope of work (Quantity of earth and soil properties) ▪ Durations of activities and their probability density functions

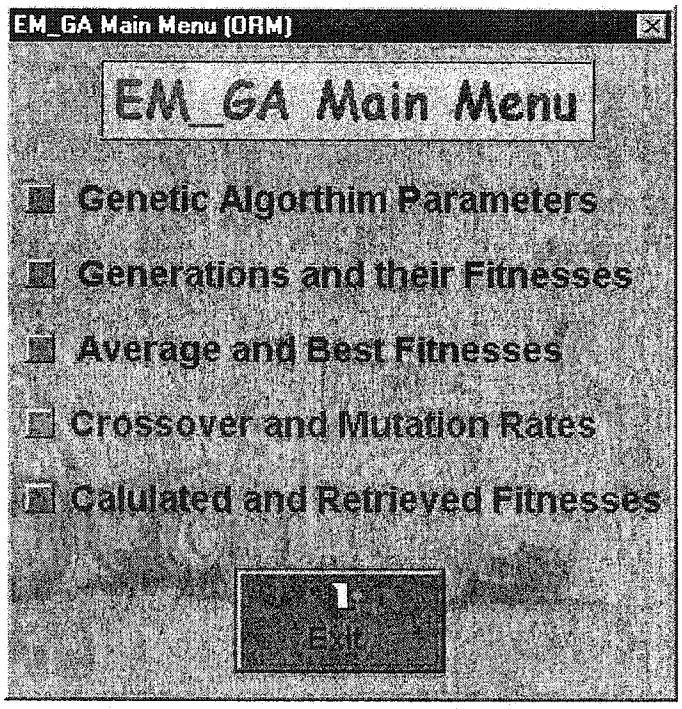


Figure 6.10 Dialog Box for EM_GA Output

The graphical reports provide charts designated to: 1) average and best fitness for the chromosomes produced in all generations, 2) crossover and mutation rates and 3) number of calculated and retrieved fitnesses. The first type of charts is used essentially to illustrate the improvement in the fitness of the generated chromosomes and to monitor the rate of conversion as the solution progresses. The second type of charts depicts the influence of the specified threshold values on the number of crossovers and mutations used throughout the solution. They could be of value when used concurrently with the first set of charts in selecting the threshold values so as to speed up the conversion process. The third type of charts illustrates the value of the developments made to speed up the computational process of the algorithm. It can be seen how quickly the number of calculated fitnesses drops continuously throughout the generation of offspring. For example, the chart shown in Figure 6.16 indicates that the algorithm activates the simulation engine (*EMSP*) 60 times in the first generation to estimate the fitnesses compared to 10 times in the ninth generation. Thus, there is a saving of 83% of the time required in estimating the fitnesses.

6.4 Numerical Example

This example considers the optimization of equipment fleets used in the earthmoving operations of a rockfill dam. This example is analyzed to illustrate the ability of the developed algorithm in selecting near-optimum fleet configurations from a set of scenarios. These scenarios, in general, represent

practical solutions that account for the availability of equipment to contractors and allow them to apply their judgment and experience in configuring these fleets. The data included in this study represent one phase of a rockfill project recently constructed in the northern part of Quebec (Hydro-Quebec 1999) as will be described later in Chapter 7. The example involves moving 3,210,000 m^3 of rock from a quarry to a rockfill dam during one of its construction phases. Rock is required to be moved from the rock quarry (RQ) to the body of the dam using the haul road (HR) shown in Figure 6.11. The properties and characteristics of borrowed rock are summarized in Table 6.2. Three fleet scenarios are considered in this example, the characteristics of their respective equipment are summarized in Table 6.3. Spread and compact activities are included, in addition to the four main activities of load, haul, dump and return. In the three fleet scenarios considered, spread and compact equipment are assumed to be of the same models to simplify the analysis (see Table 6.4). The ranges for the number of equipment involved were set around the user-defined fleet configurations listed in Table 6.5. And the activity durations data is summarized in Table 6.6. Upon defining the individual fleet scenarios and necessary data for the developed algorithm (see Figure 6.9 and Table 6.7), *EM_GA* was activated.

Table 6.2 Rock Properties

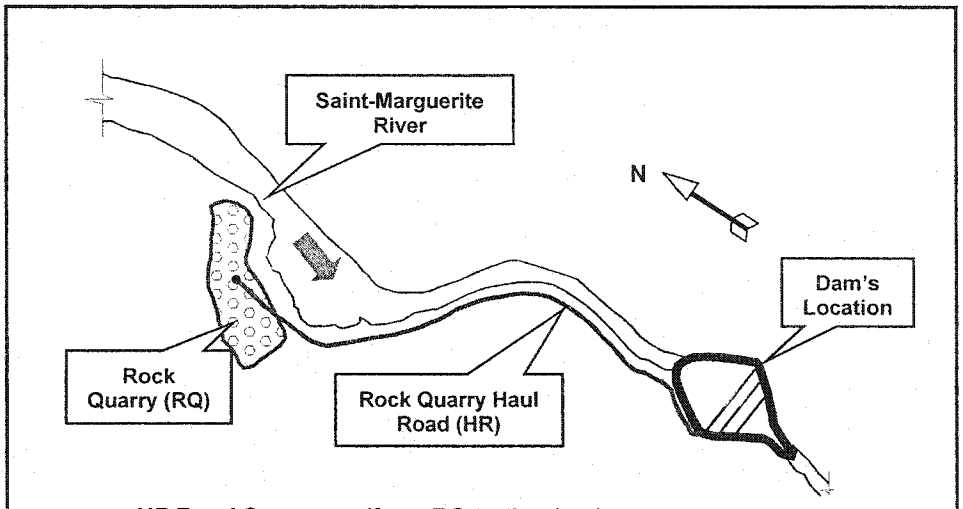
Bank Density (Kg/ m^3) :	2,730
Loose Density (Kg/ m^3) :	1,660
Bucket fill factor (%) :	80

Table 6.3 Characteristics of Fleet Scenarios

F_S1	
Loaders (Wheel Type)	
Type :	CAT 992G
Defined Range (min no.-max no.) :	2-8
Bucket Capacity (m^3):	12.3
No of passes:	5
Hourly Owning and Operating Cost ($/hr):	295.74
Haulers (Off-highway Truck)	
Type:	CAT 777D
Defined Range (min no.-max no.) :	10-60
Payload (ton):	81.7
Hourly Owning and Operating Cost ($/hr):	212.95
F_S2	
Loaders (Wheel Type)	
Type:	CAT 990SII
Defined Range (min no.-max no.) :	2-8
Bucket Capacity (m^3):	9.2
No of passes :	4
Hourly Owning and Operating Cost ($/hr) :	243.35
Haulers (Off-highway Truck)	
Type:	CAT 773D
Defined Range (min no.-max no.) :	10-60
Payload (ton):	48.9
Hourly Owning and Operating Cost ($/hr):	160.53
F_S3	
Loaders (Wheel Type)	
Type:	CAT 988F
Defined Range (min no.-max no.) :	2-8
Bucket Capacity (m^3) :	6.3
No of passes :	4
Hourly Owning and Operating Cost ($/hr) :	174.21
Haulers (Articulated Truck)	
Type:	CAT D400D
Defined Range (min no.-max no.) :	10-60
Payload (ton):	33.46
Hourly Owning and Operating Cost ($/hr):	134.33

Table 6.4 Characteristics of Spread and Compact Equipment

Spread Equipment (Dozer)	
Type :	CAT D8R
Defined Range (min no.-max no.) :	1-8
Cycle Production (m^3):	27
Hourly Owning and Operating Cost ($/hr):	153.51
Soil Compactor	
Type:	CAT CS-583C
Defined Range (min no.-max no.) :	1-8
Cycle Production (m^3):	19.1
Hourly Owning and Operating Cost ($/hr):	89.62



HR Road Segments (from RQ to the dam)

Seg.	*Length (m)*	*RR (%)*	*Grade (%)*
1	457	4	-2.8
2	787	2	0.1
3	710	2	-0.2
4	955	2	3.3
5	185	2	0
	3,094		

Figure 6.11 Quarry Location and Haul Road Characteristics

Table 6.5 User-Defined Fleet Configurations

Fleet	Load	Haul	Spread	Compact
F_S1	4	30	3	3
F_S2	3	35	3	4
F_S3	4	40	3	4

Table 6.6 Activity Time Distributions

Fleet	Load	Haul	Dump	Return	Spread	Compact
F_S1	T(3.94, 4.57, 4.15)	T(4.3, 4.53, 4.98)	U(1.9, 2.2)	T(3.17, 3.34, 3.67)	T(2.47, 2.6, 2.86)	T(1.8, 1.9, 2.09)
F_S2	T(3.15, 3.32, 3.65)	T(4.17, 4.39, 4.83)	U(1.7, 1.9)	T(3.1, 3.26, 3.59)	T(2.47, 2.6, 2.86)	T(1.8, 1.9, 2.09)
F_S3	T(3.15, 3.32, 3.65)	T(4.59, 4.83, 5.31)	U(1.4, 1.6)	T(3.45, 3.73, 4.1)	T(2.47, 2.6, 2.86)	T(1.8, 1.9, 2.09)

Note:
U (N1, N2); U : Uniform Distribution, N1: Lower Limit and N2: Upper Limit
T (N1, N2, N3); T : Triangle Distribution, N1: Lower Limit, N2: Mode and N3: Upper Limit

Table 6.7 EM_GA Parameters

Sub-population size:	20
No. of generations:	100
No. of pilot simulation:	5
Non-improvement limit:	20
Crossover probability:	0.70
Mutation probability:	0.05
Elitism:	Applicable
Extreme fleet configurations:	Applicable
User-defined configurations:	Applicable
Scheduled daily hours:	16
No. of working days per month:	25
Time-related indirect cost ($/month):	500,000
Time-independent indirect cost($):	1,000,000

Three scenarios were considered; each is represented by a sub-population of 20 chromosomes. As such, a total of 60 fleet configurations were created in each new generation produced by the developed algorithm. The algorithm yielded a near-optimum fleet (FL_GA1), configured from the first fleet scenario (F_S1). The number of each type of equipment in that fleet is shown in Figure 6.12 and Table 6.8. Different charts have been generated for FL_GA1 as shown in Figures (6.13, 6.14, 6.15, and 6.16). Subsequently, the simulation analysis was performed for the recommended fleet configuration using *EMSP*. The analysis results indicated the project total duration and total cost to be 1,231 hrs and $13,964,212, respectively.

In view of the randomness inherent in the generation process of fleet configurations (i.e. chromosomes) in the developed algorithm, it is recommended

that users experiment with a set of runs, similar to that described above, for overall assessment and evaluation of results obtained. In this regard, a second run was performed and a new near-optimum fleet was obtained (FL_GA2), configured from the second fleet scenario (F_S2). The total project duration and cost were found to be 1,436 hrs and $16,143,609, respectively. It should be noted that the number of populations generated for FL_GA2 was 38, which indicates that the best fitness was obtained after 19 generations (see Figure 6.17). The analysis results are summarized in Table 6.8 and Figure 6.18.

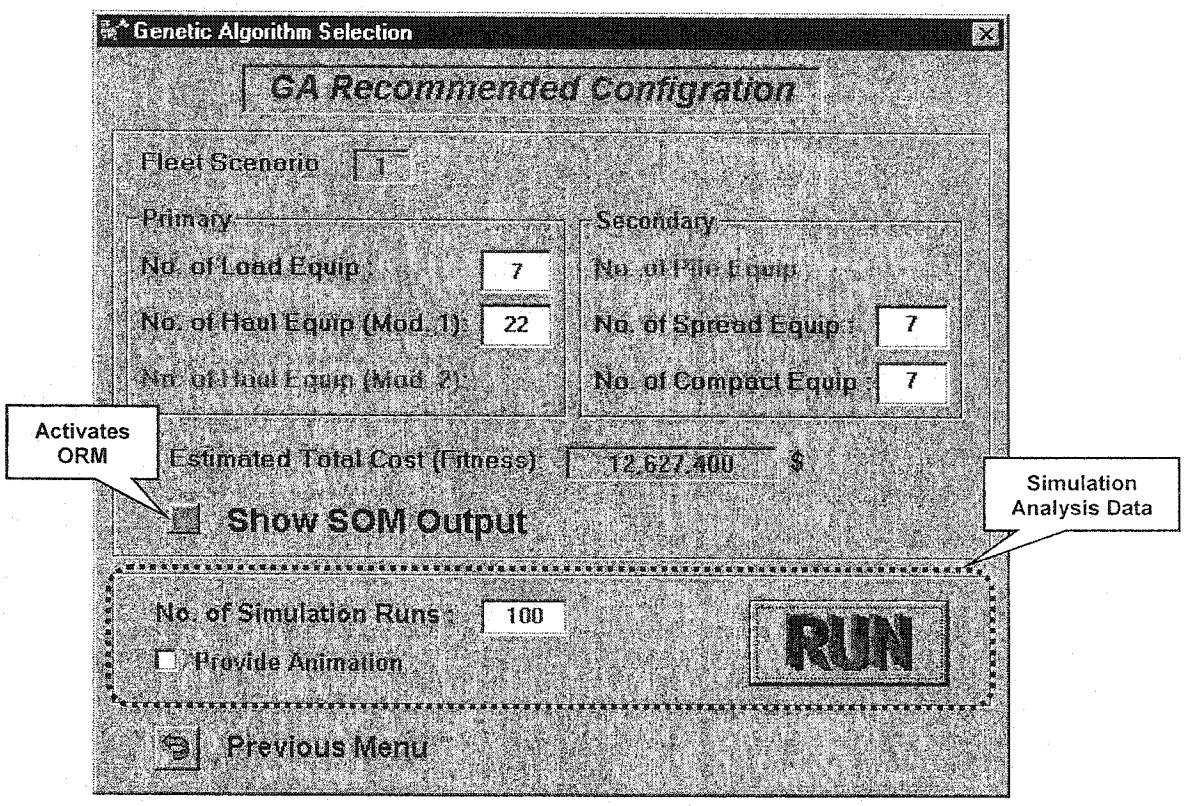


Figure 6.12 Recommended Fleet Configuration Dialog Box

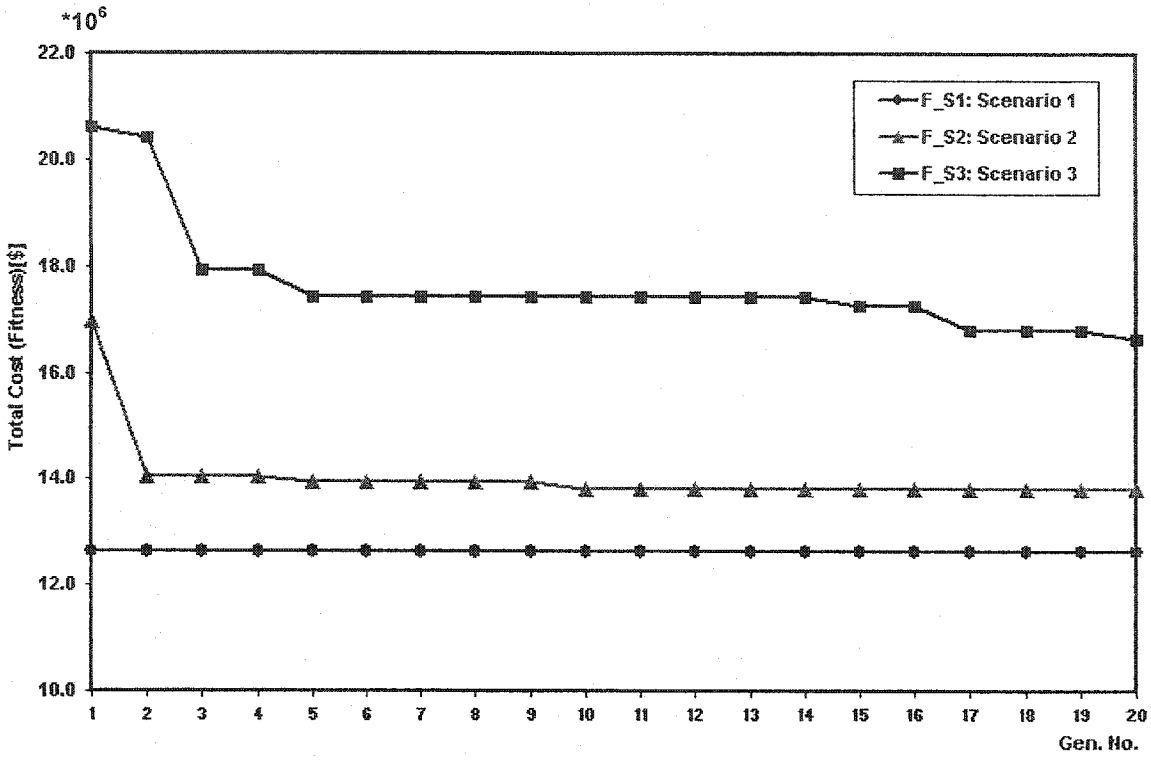


Figure 6.13 Best Fitnesses of Generations (FL_GA1 Run)

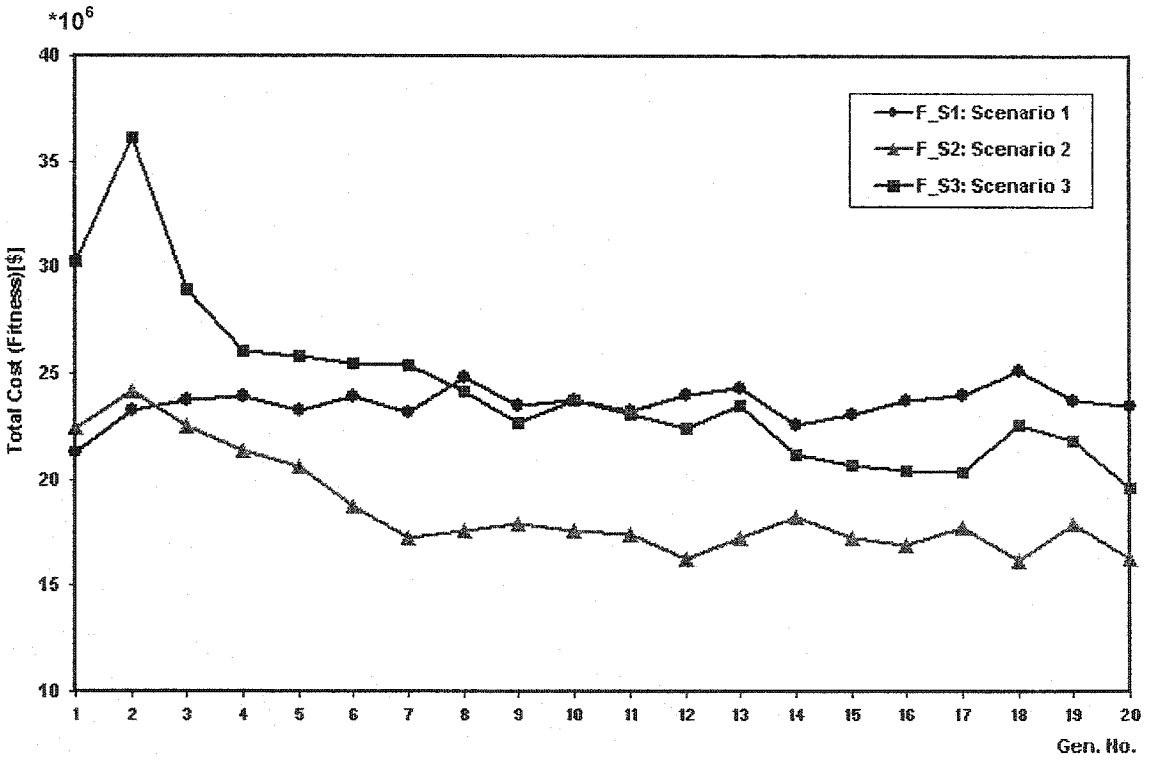


Figure 6.14 Average Fitnesses of Generations (FL_GA1 Run)

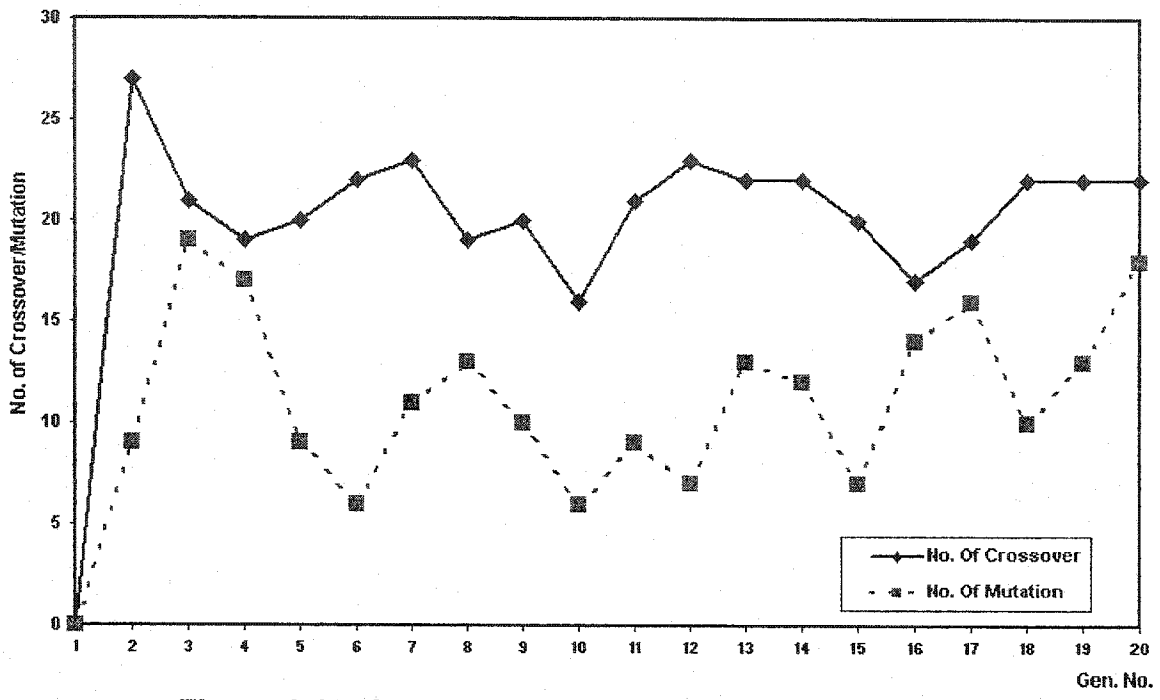


Figure 6.15 Crossovers and Mutations (FL_GA1 Run)

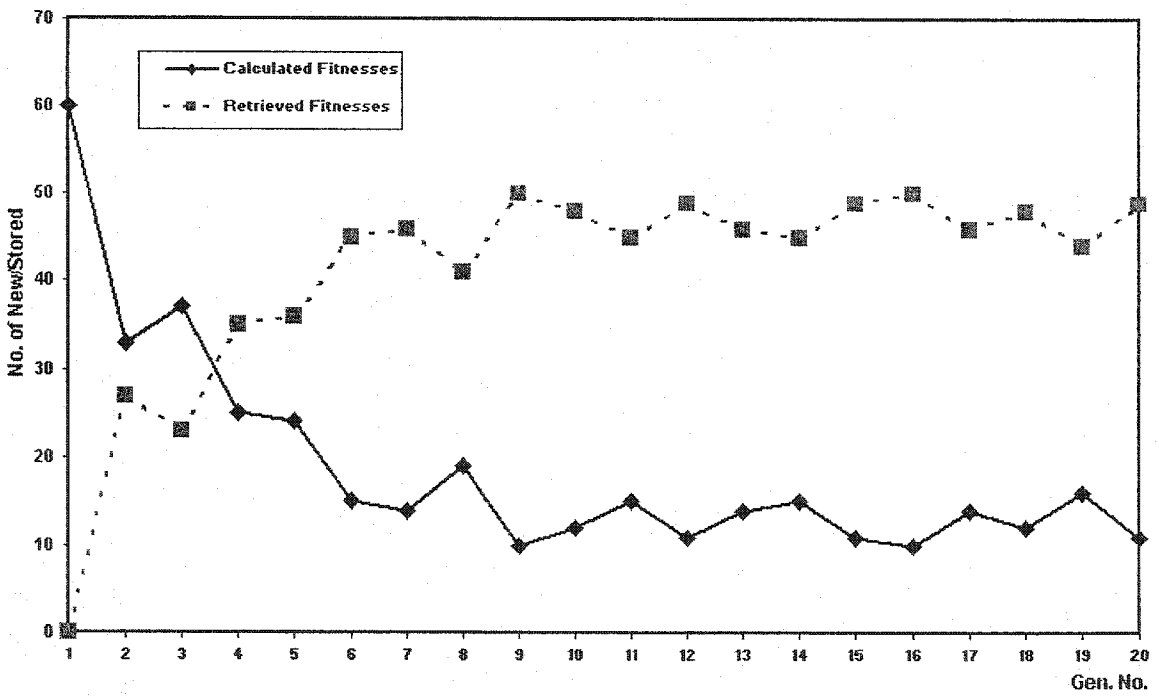


Figure 6.16 Calculated and Retrieved Fitnesses (FL_GA1 Run)

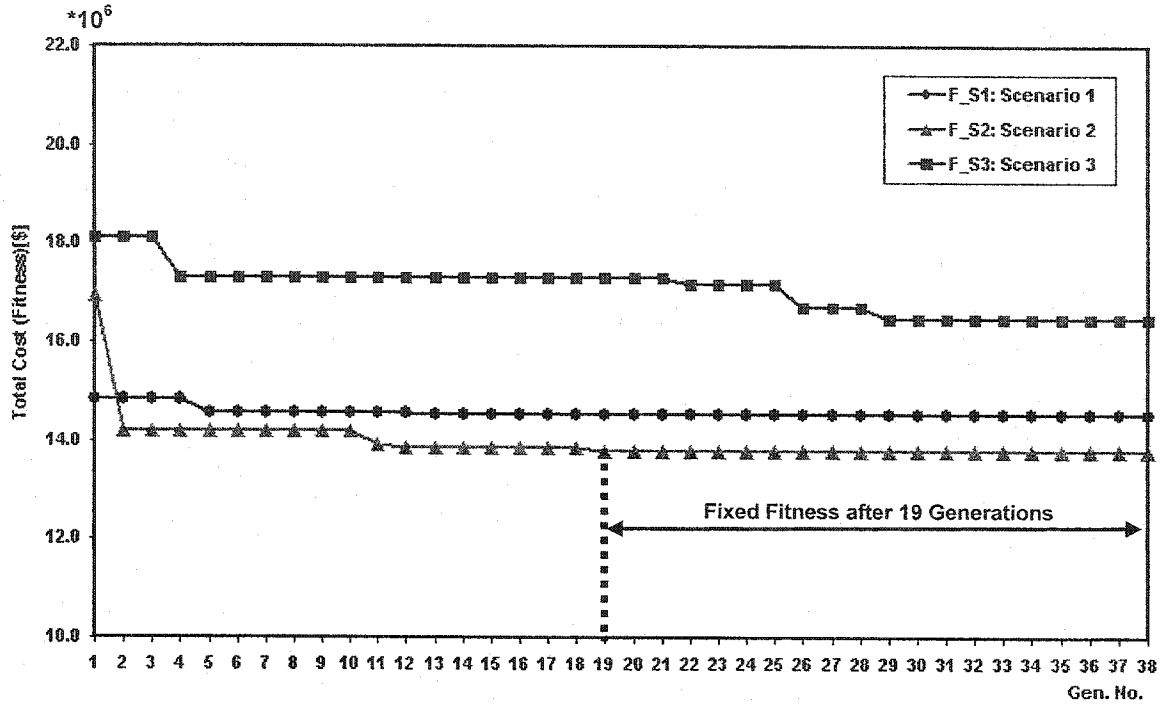


Figure 6.17 Best Fitnesses of Generations (FL_GA2 Run)

Table 6.8 Selected vs. User-Defined Fleets

Fleet Name	Fleet Scenario	Fleet Configuration	Total Duration (Hrs.)	Total Cost ($)	Note
FL_1	F_S1	(2,10,1,1)	8864	26,497,940	Lower-Fleet Limit
FL_2	F_S1	(8,60,8,8)	1109	22,747,567	Upper-Fleet Limit
FL_3	F_S1	(4,30,3,3)	2955	24,965,229	User-Configured
FL_4	F_S2	(2,10,1,1)	8865	27,612,292	Lower-Fleet Limit
FL_5	F_S2	(8,60,8,8)	1257	24,923,479	Upper-Fleet Limit
FL_6	F_S2	(3,35,3,4)	3351	37,927,260	User-Configured
FL_7	F_S3	(2,10,1,1)	8865	28,524,994	Lower-Fleet Limit
FL_8	F_S3	(8,60,8,8)	1835	27,927,163	Upper-Fleet Limit
FL_9	F_S3	(4,40,3,4)	3670	35,962,817	User-Configured
FL_GA1	F_S1	(7,22,7,7)	1231	13,964,212	Selected by EM_GA
FL_GA2	F_S2	(8,29,6,7)	1436	16,143,609	Selected by EM_GA

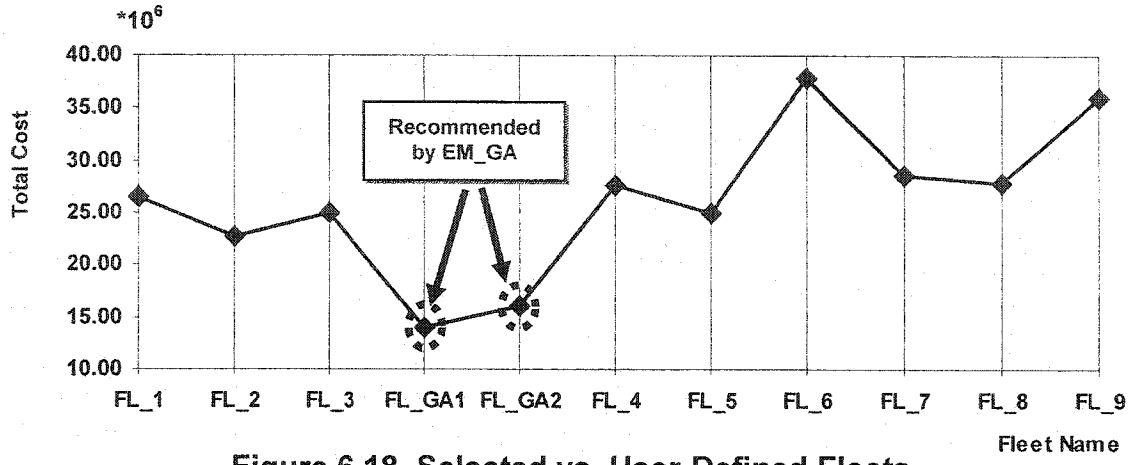


Figure 6.18 Selected vs. User-Defined Fleets

6.5 Summary

This chapter presented a methodology for simulation optimization utilizing Genetic Algorithms (GAs). The methodology was implemented in a prototype software (*EM_GA*) that runs within the developed system (*SimEarth*). *EM_GA* accounts for both qualitative and quantitative variables that influence the production of earthmoving operations. It also considers for the indirect cost portion associated with those operations. The chapter focused on the development of the algorithm and the description of its basic features including crossover, mutation and elitism. The developed algorithm accounts for the consideration of practical fleet configurations that account for equipment availability. It also has an open architecture that allows contractors to use their judgment and experience in defining the initial set of fleet scenarios including the number of equipment used in each one.

EM_GA has a powerful computational utility that speeds up the calculations by employing elitism and by storing the fitnesses of chromosomes in a built-up database to avoid duplication of fitness calculation. The proposed algorithm was implemented utilizing MS Visual C++ 6.0 and its dialog boxes were coded in MS Visual Basic 6.0. A numerical example was presented to demonstrate the different features of the developed algorithm and to illustrate its capabilities in selecting near-optimum fleets that minimize total project cost.

CHAPTER 7

IMPLEMENTATION OF THE PROPOSED SYSTEM: SIMEARTH

7.1 General

This chapter presents the implementation of the developed *SimEarth* system. Except for the dynamic sub-module, all the components of the developed system have been implemented in MS environment to facilitate integration (see Figure 7.1). *Proof* Animation (2000) software has been used in the development of the dynamic sub-module. All user interface screens have been designed and implemented utilizing Visual Basic 6.0. The system runs on Microsoft Windows 95, 98, 2000 or NT.

7.2 SimEarth User Interfaces

Forty-one user interfaces "*Forms*" have been designed and coded to facilitate: 1) entering project data (e.g. scope of work, fleet configuration, activity durations, simulation characteristics, etc.); 2) storing equipment data, 3) exporting and importing data to/from external output files, 4) activation of *SimEarth*'s components and their respective functions and algorithms; and 5) generation output reports. In addition to these *Forms*, three "*Modules*" have been coded to store *SimEarth*'s variables and functions as "*Public*" ones in order to be accessible within the developed *SimEarth*. Figures 7.2 and 7.3 depict the designated "*Forms*" and "*Modules*" of *SimEarth* and the first user interface.

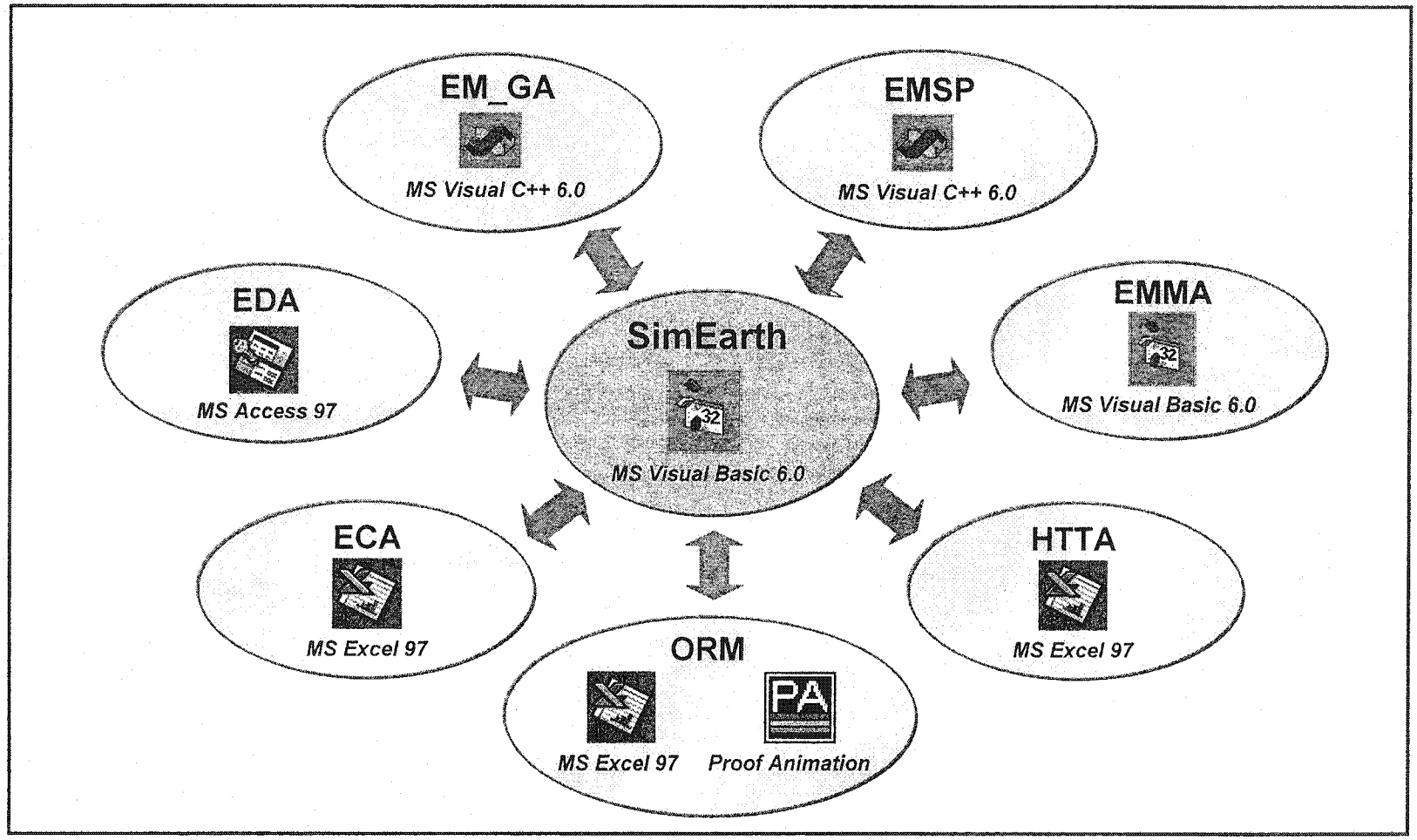


Figure 7.1 Integration of SimEarth Components

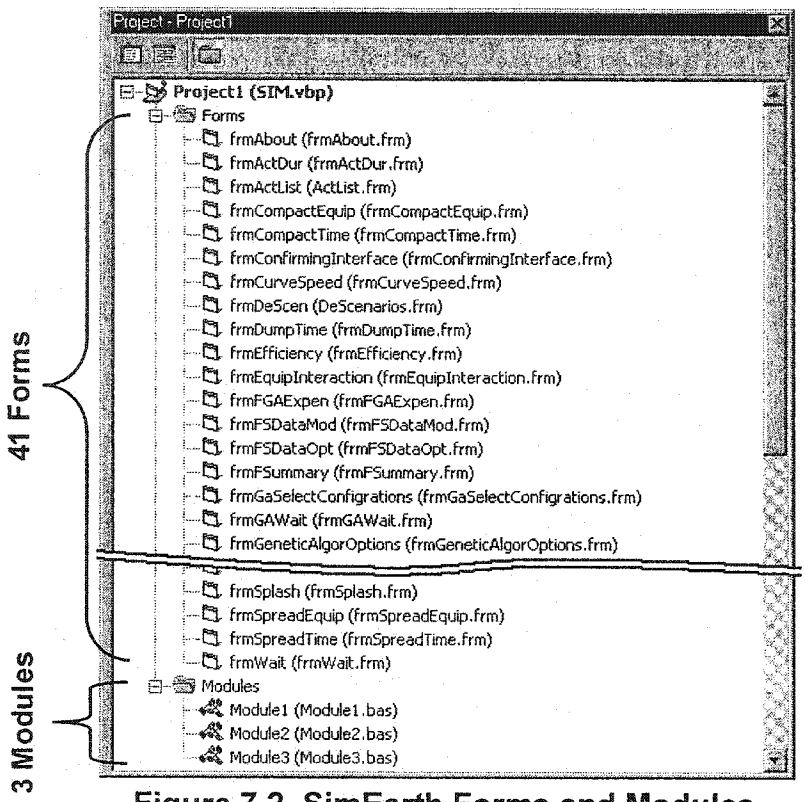


Figure 7.2 SimEarth Forms and Modules

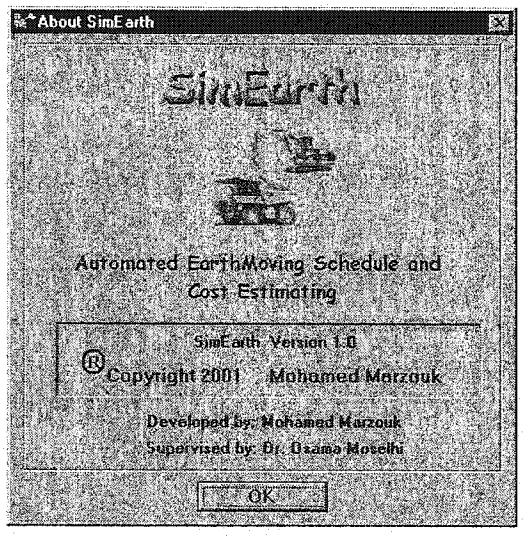


Figure 7.3 SimEarth Initial User Interface

SimEarth is an automated environment that accounts for different aspects of earthmoving projects including their phases and quality assurance requirements. For the project phase, the user can divide the project into stages (e.g. initial

build-up, normal and run-down). The characteristics (i.e. scope of work, utilized equipment, etc.) of each phase and subsequently used to perform simulation analysis for that phase. On the other hand, quality assurance requirements for certain work activities can be accounted for implicitly in their respective durations and associated costs.

Upon initial activation of the system, the menu depicted in Figure 7.4 is activated. This menu provides the user with four alternatives: 1) activate equipment cost application (*ECA*); 2) activate equipment database application (*EDA*); 3) activate earthmoving simulation program (*EMSP*); or 4) exit *SimEarth*.

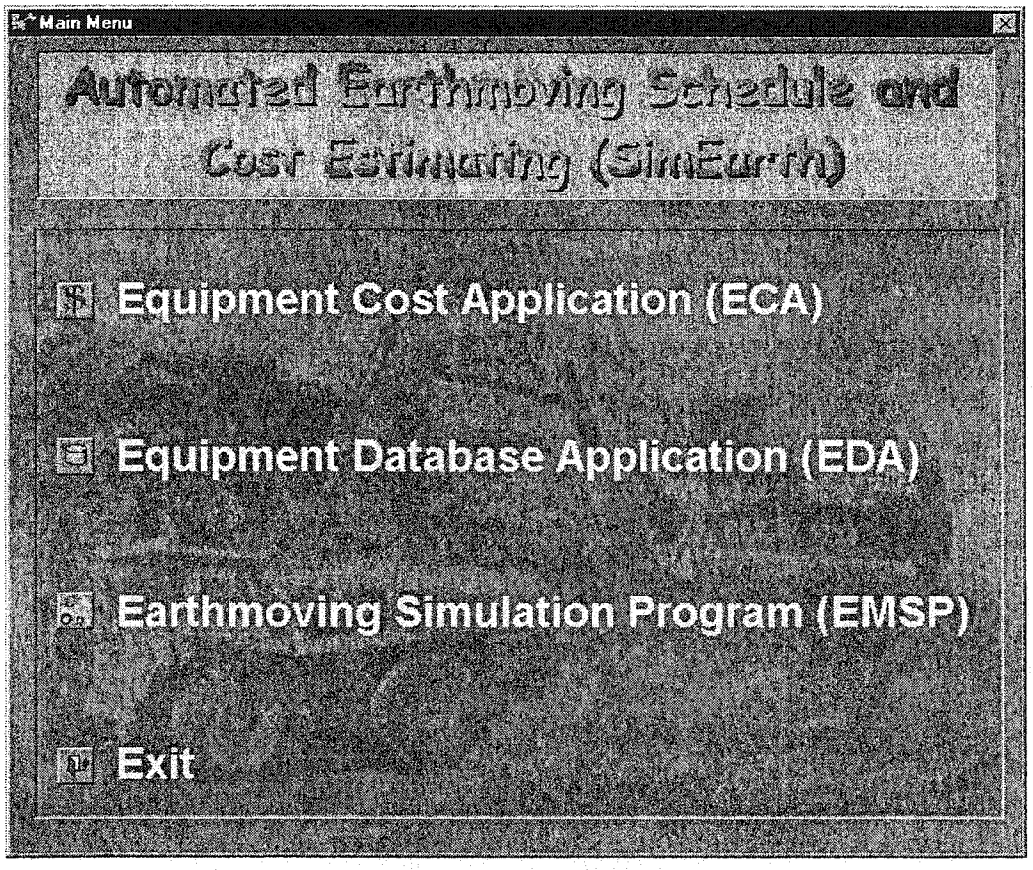


Figure 7.4 SimEarth Main User Interface

Selecting the *EMSP* actives project information screens as depicted in Figure 7.5. These screens allow the user to: 1) define project characteristics (e.g. its name, scheduled hours/day, and scheduled days/month); 2) enable the activation of the genetic algorithm (*EM_GA*); 3) define the indirect cost components of the project at hand; and 4) define project data (e.g. scope of work, road segment characteristics, fleet configurations) and run simulation analysis. The following subsections provide an overview of the designated user interfaces designed to facilitate the use of *SimEarth*.

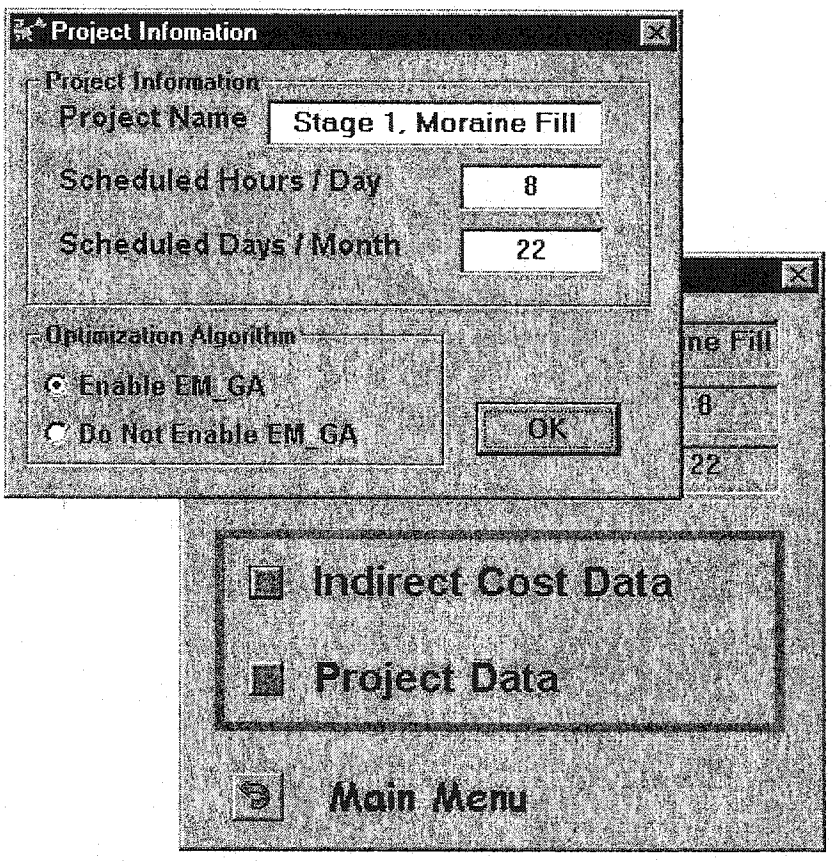


Figure 7.5 SimEarth Project Information User Interface

7.2.1 Project Indirect Cost

Indirect cost of earthmoving operations are sizeable compared to building construction and can exceed direct cost. An effort has been made to prepare realistic estimates of those costs, considering the cost components proposed by Hicks (1992) and the input received from the interviews with earthmoving experts. The cost components, considered in the developed system, are grouped in three categories: 1) field support cost; 2) field general and administrative expenses; and 3) field summary cost. These components are either time-related or time-independent (i.e. in the form of a lump sum). Table 7.1 lists the components within each category used in *SimEarth*. This list serves as a menu where the user can select from, in an interactive manner, the cost components relevant to the project under study (see Figures 7.6-a, 7.6-b, 7.6-c and 7.6-d). In addition, the user can further define additional component(s) or enter the amount of subcontracted work as a lump sum in the designated dialog boxes.

An efficient way of modeling indirect cost is to consider the equipment plant as a profit center, whereby its superintendent establishes the hourly total cost of each piece of equipment. The project, then, rents needed pieces of equipment from the plant based on their established hourly cost. This creates a competitive environment where the project team makes the best effort to efficiently utilize the equipment it pays for. It also helps the plant team to be competitive with outside equipment suppliers and ultimately leads to an efficient and well-managed project.

Table 7.1 Indirect Costs Default Components

Category/Sub Category	Payment Type	Component
1. Field support cost		
1.1 Mobilization	Lump Sum	1.1.1 Equipment
1.2 Equipment erection	Lump Sum	1.2.1 Shovels
1.3 Plant installation	Lump Sum	1.3.1 Equipment shops
		1.3.2 Field office and yard
		1.3.3 Roadway
		1.3.4 Power distribution
		1.3.5 Craft shops
		1.3.6 Warehouses
		1.3.7 Construction camp
1.4 Plant operation	/Month	1.4.1 Equipment shops
		1.4.2 Warehouses
		1.4.3 Road maintenance
2. Field general and administrative	/Month	2.1 Project managers
		2.2 Field engineers
		2.3 Field supervisors
		2.4 Accountants and controllers
		2.5 Project offices
		2.6 Field services
		2.7 Field layout
		2.8 Quality control
3. Field summary cost	Lump Sum	3.1 Guaranteed wages
		3.2 Sales taxes
		3.3 Special insurance
		3.4 Bonus or damages

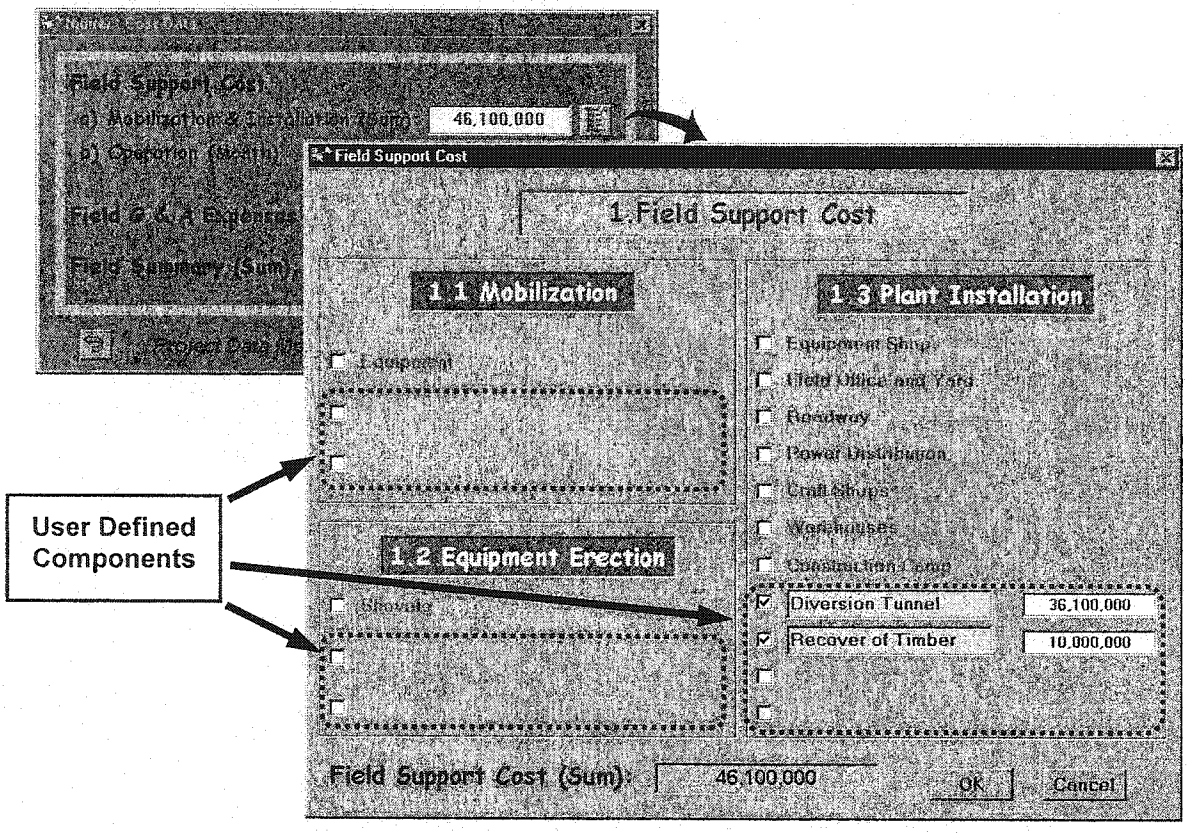


Figure 7.6-a Field Support Components (Lump Sum)

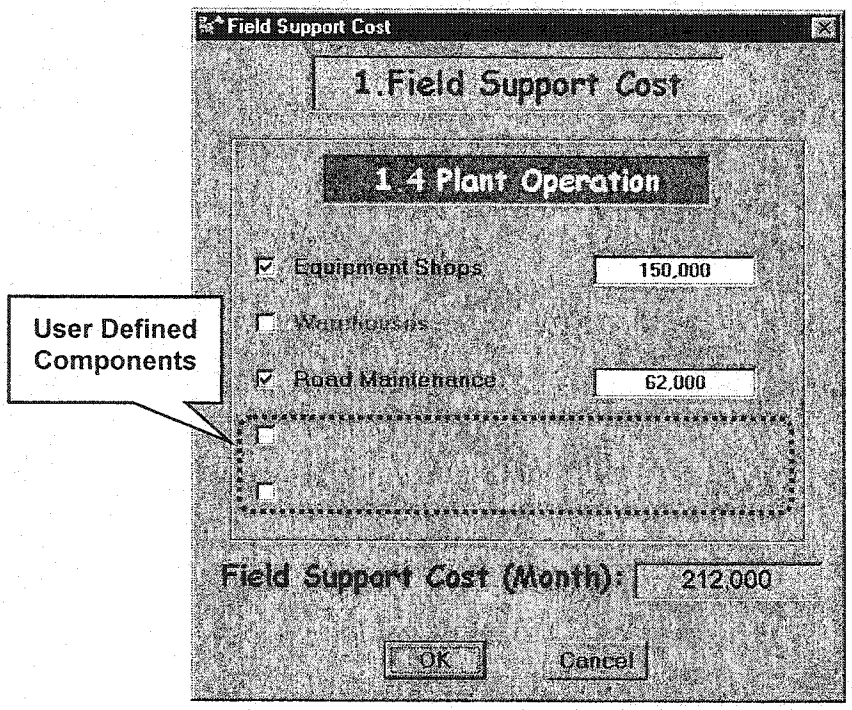


Figure 7.6-b Field Support Components (Monthly)

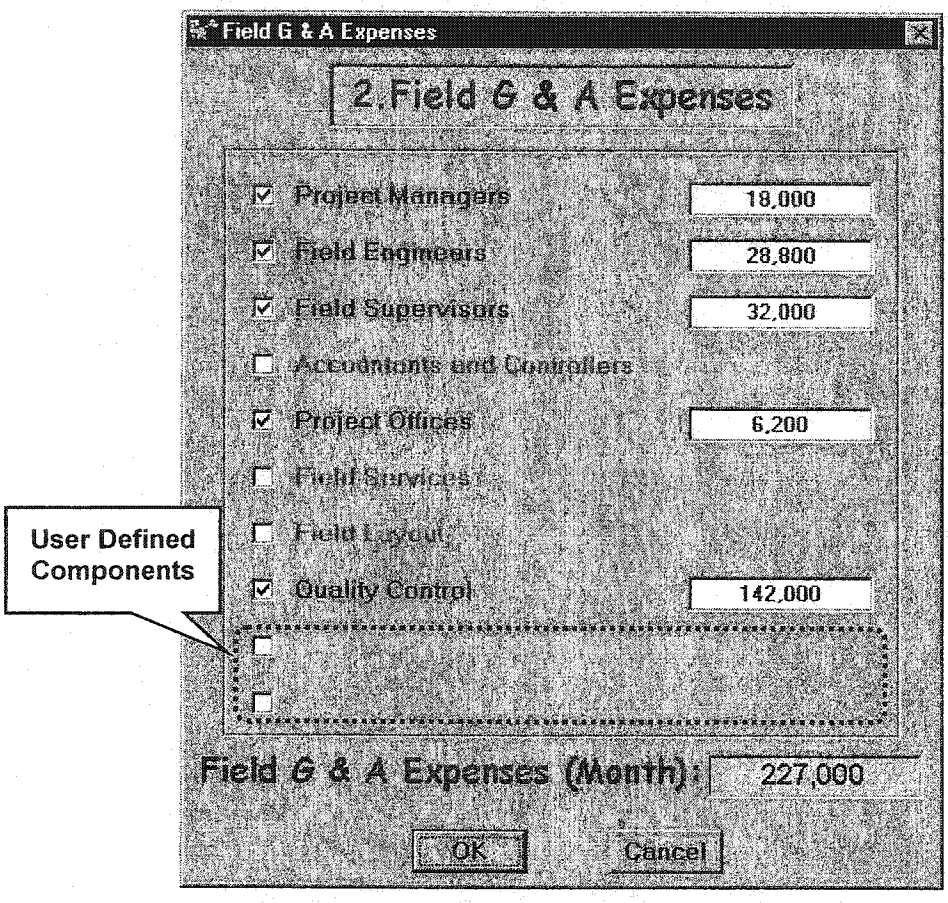


Figure 7.6-c Field General and Administrative Components (Monthly)

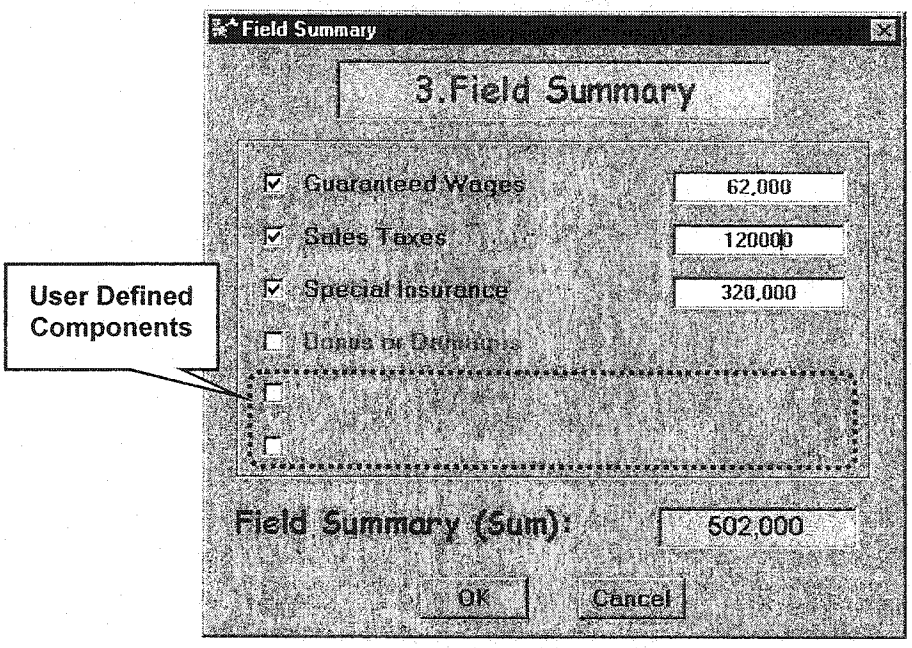


Figure 7.6-d Field Summary Components (Lump Sum)

7.2.2 Soil Quantities and Characteristics

The scope of work is defined in order to set the termination condition of the simulation analysis. Different soil characteristics are defined: loose density,, bank density, water content percent and bucket fill factor. The user can activate the appropriate function to retrieve loose and bank densities of various soils from the designated database (see Figure 7.7).

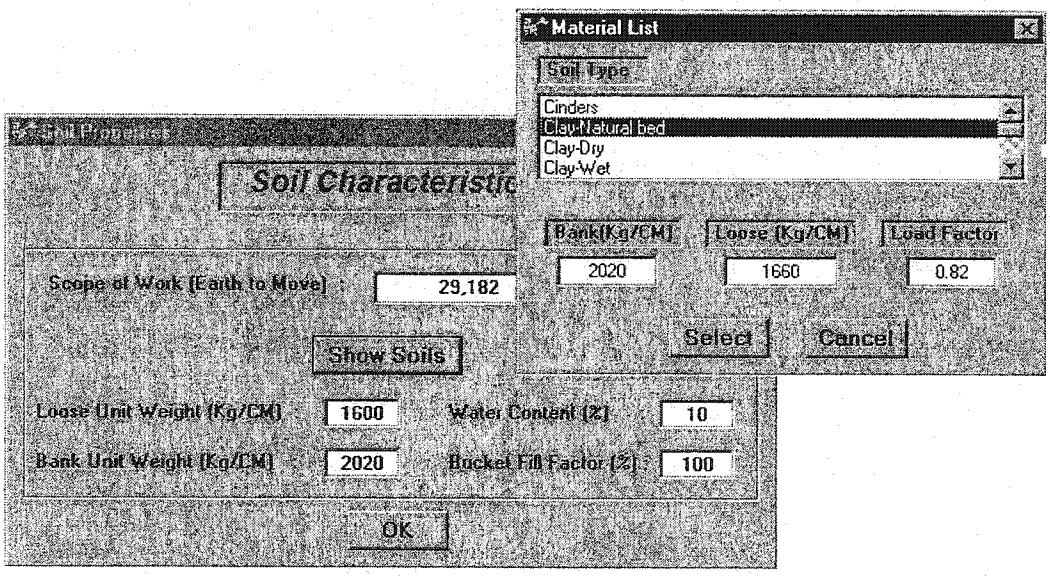


Figure 7.7 Soil Quantities and Characteristics

7.2.3 Haul Road Characteristics

The characteristics of the haul road such as: 1) number of haul road segments, 2) number of return road segments or mirror haul road segments, 3) length of each segment, 4) grade resistance percent, 5) rolling resistance percent and 6) maximum allowable speed are entered by making use of the interactive screens shown in Figure 7.8. This data is used by *HTTA* to estimate haulers' travel time.

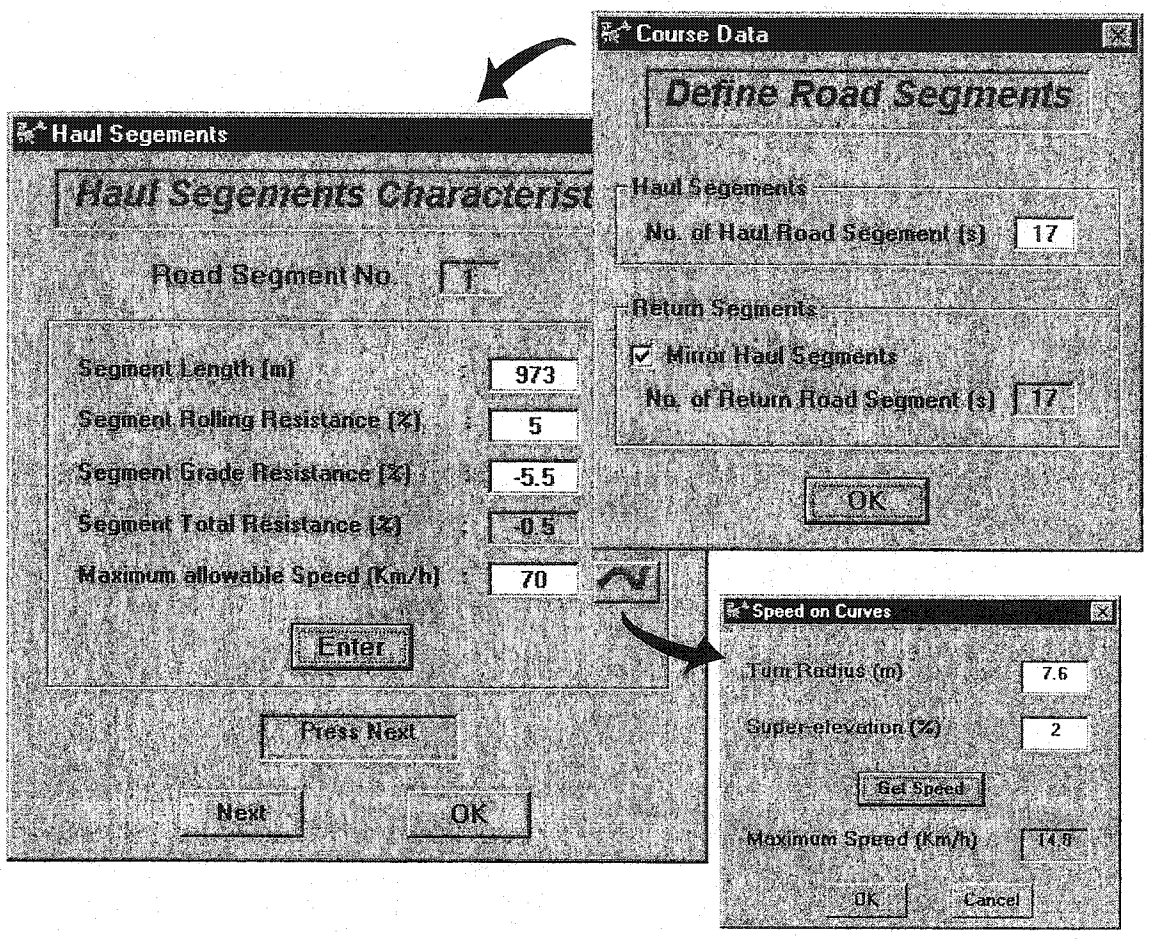


Figure 7.8 Characteristics of Haul Road Segments

7.2.4 Fleet Configurations

Fleet configurations are retrieved from *EDA* and fed into *SimEarth*. This is carried out by determining the number of fleet scenarios along with the secondary activities (pile, spread and compact); and the main activities (load, haul, dump and return). Accordingly, five types of earthmoving equipment are stored in the developed system (see Figure 7.9). These are: 1) load, 2) haul, 3) pile, 4) spread, and 5) compact equipment.

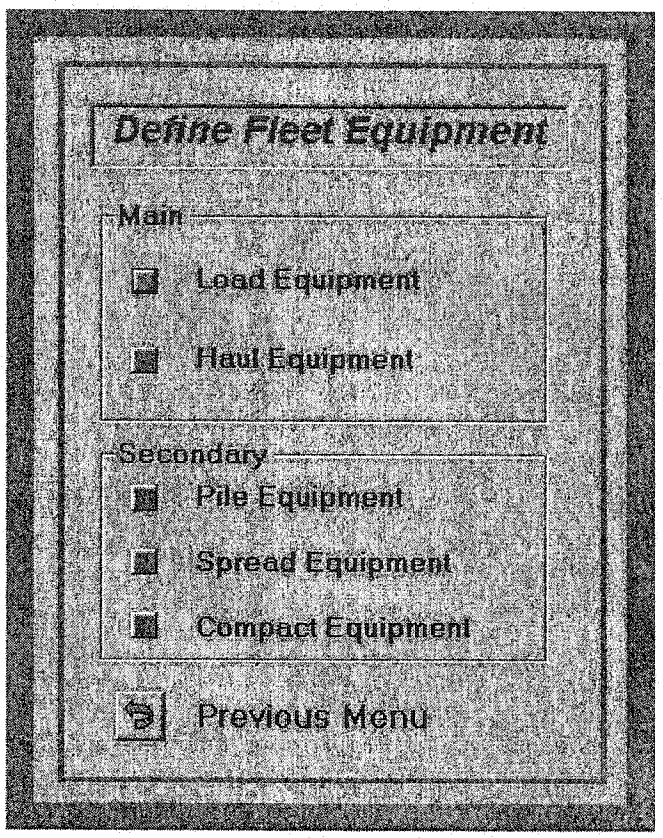


Figure 7.9 Fleet Equipment User Interface

Load equipment is defined by specifying the number of loaders and their related data. The number of loaders includes: upper limit, lower limit and user-defined (which indicates the user preference for the project at hand). The related data include: type, model, bucket capacity and hourly owing and operating costs. The user can select load equipment type from the four available types displayed in the *Load Equipment* dialog box. *SimEarth* stores the models and bucket capacities that correspond to each type of equipment (see Figure 7.10). Three user interfaces have been designed to facilitate data entry for standard cycle time, efficiency, and matched hauler (see Figure 7.10).

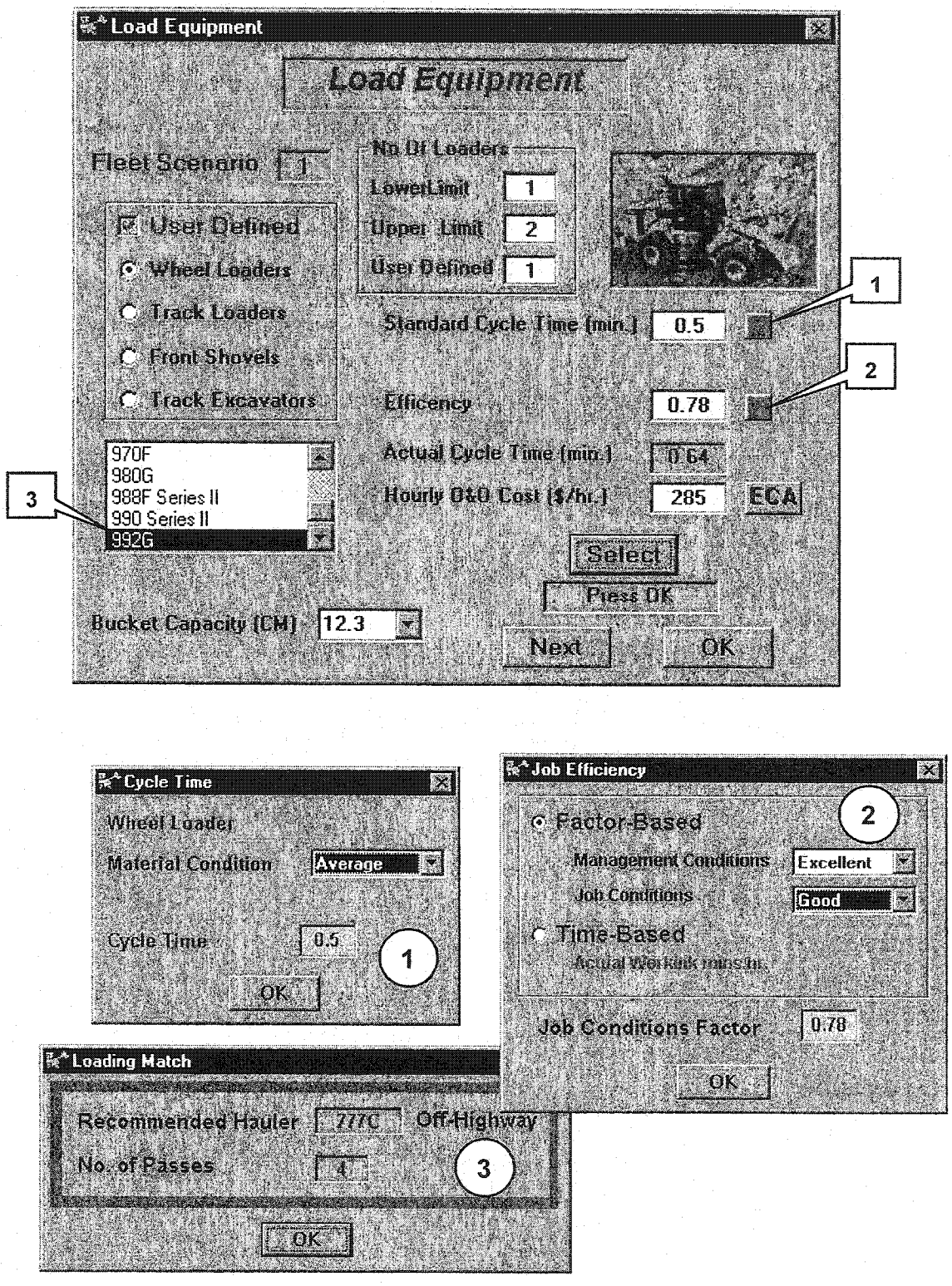


Figure 7.10 Load Equipment User Interface

SimEarth allows the use of two models of haul equipment within a fleet scenario. There are six available haulers displayed in the *Haul Equipment* dialog box (see Figure 7.11). For construction and mining trucks and articulated trucks, the user is required to define the number of passes performed by loaders.

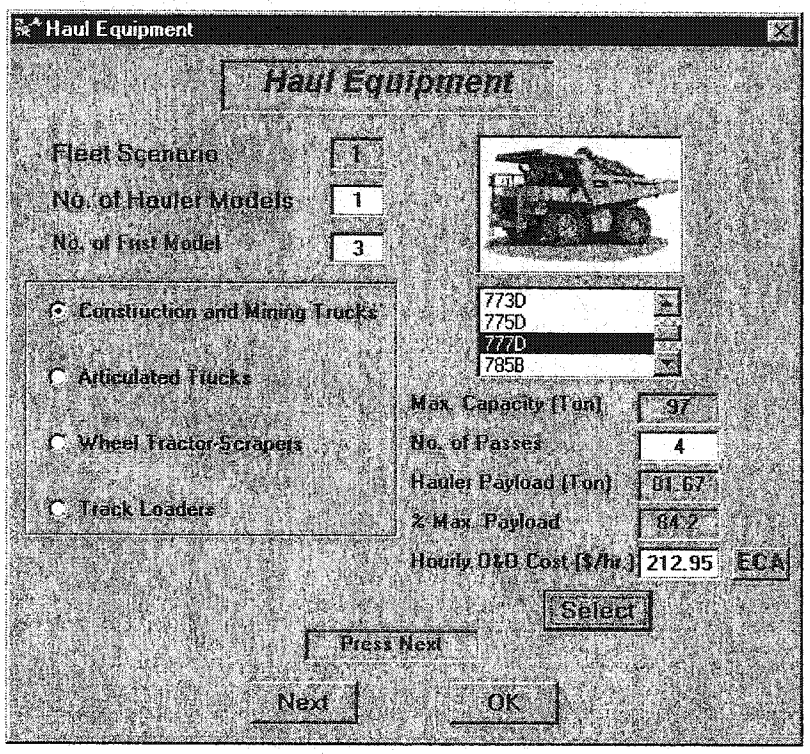


Figure 7.11 Haul Equipment User Interface

Based on their respective bucket capacity, bucket fill factor and the number of passes, *SimEarth* estimates the hauler's payload and percent of maximum capacity as follows:

$$PI = \gamma_L * BC * N * F_B \qquad (7.1)$$

$$\%PI_{max} = PI/MC \qquad (7.2)$$

In which : PI is the hauler's payload (ton); γ_L is the loose density (ton/m^3); BC is bucket capacity (m^3); N is the number of passes; F_B bucket fill factor; $\%PI_{max}$ is the percent of the maximum payload; and MC is the maximum capacity (ton). Similarly, user interfaces have been designed and coded to facilitate selection of pile, spread, and compact equipment from a set of available equipment. Figures 7.12, 7.13 and 7.14 show the designated user interfaces for of pile, spread, and compact equipment, respectively.

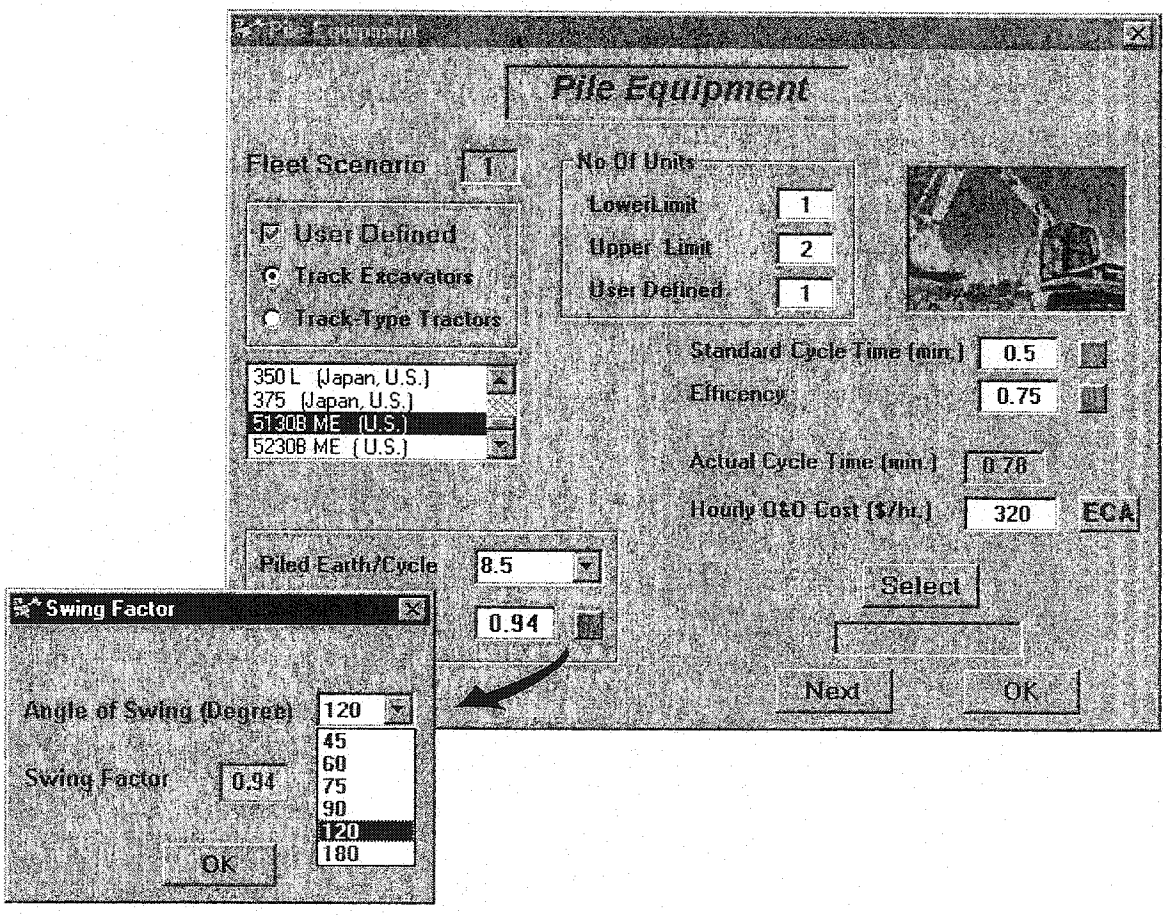


Figure 7.12 Pile Equipment User Interface

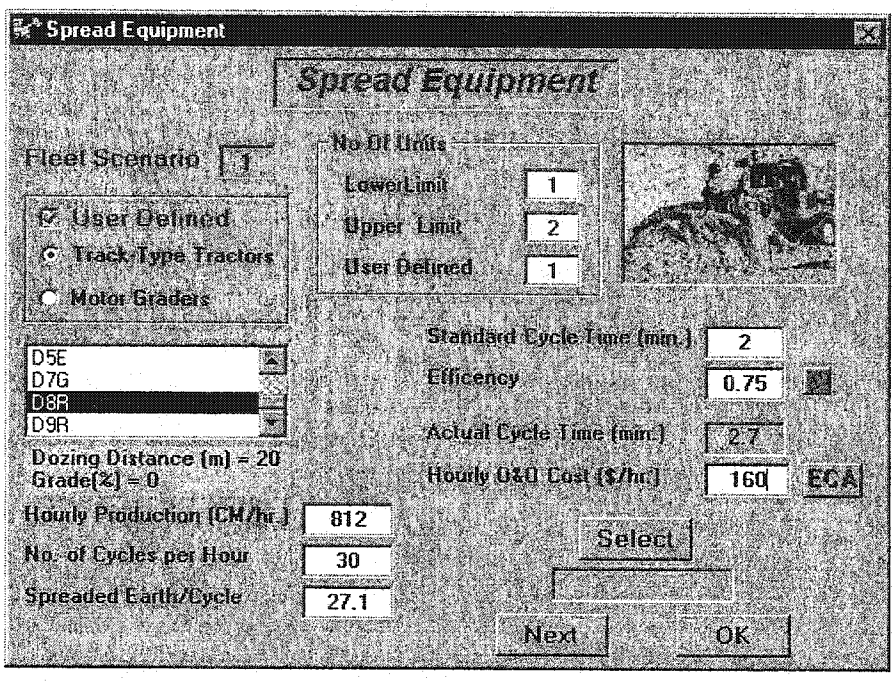


Figure 7.13 Spread Equipment User Interface

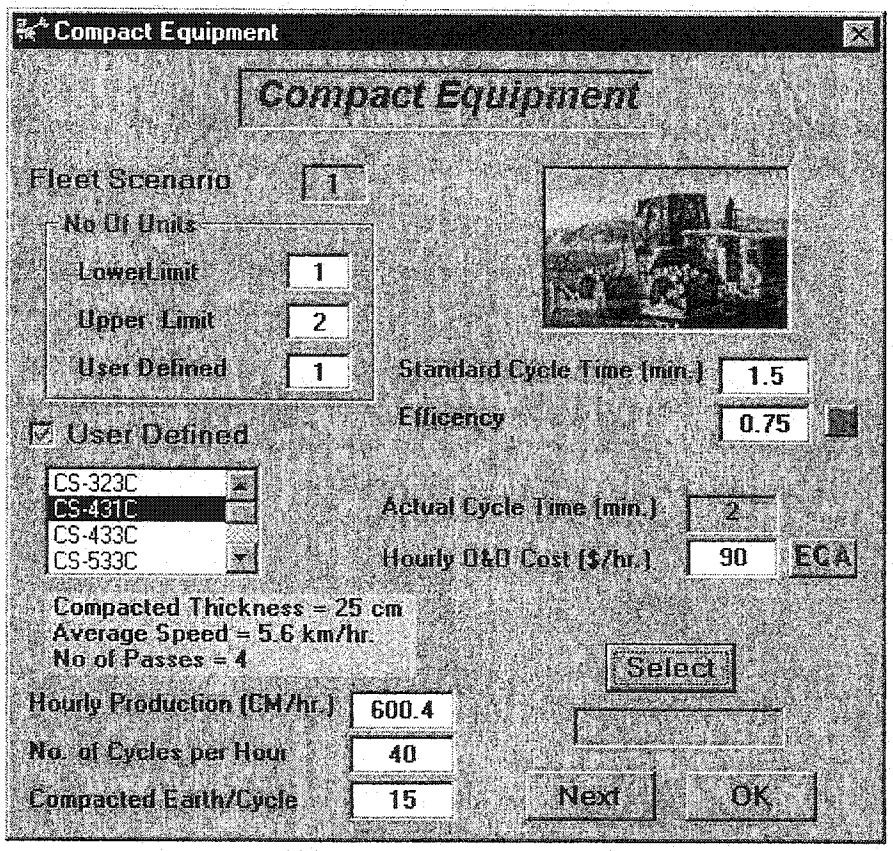


Figure 7.14 Compact Equipment User Interface

7.2.5 Activity Durations

The user of *SimEarth* defines the duration of its seven activities (main and secondary) through the user interface shown in Figure 7.15. These activities are then assigned equipment and probability density functions to model their stochastic durations. Eight distributions are coded in the proposed system: 1) uniform, 2) triangle, 3) normal, 4) exponential, 5) erlang, 6) lognormal, 7) gamma, and 8) beta distributions (Marzouk and Moselhi 2001). The mathematical modeling of these distributions follows Prisker (1997) and their code is included in Appendix A. The user can select the most appropriate distribution for the task at hand, otherwise the system defaults to a triangular distribution.

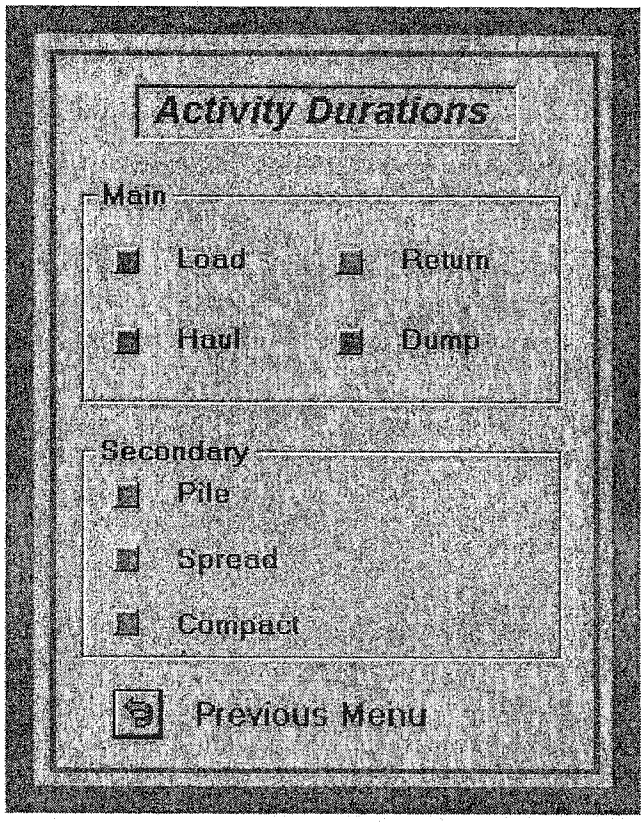


Figure 7.15 Activity Durations User Interface

For example, for a haul trip time that follows the default triangular distribution, the estimated travel time calculated using *HTTA* is used as the most likely value for that distribution. Default values are set for optimistic and pessimistic time making use of the calculated travel time. A reduction of 5% and increase of 10% with respect to the estimated travel time are used for that purpose. Alternatively, the user can enter his/her estimate for optimistic and pessimistic travel time using the input screen shown in Figure 7.16.

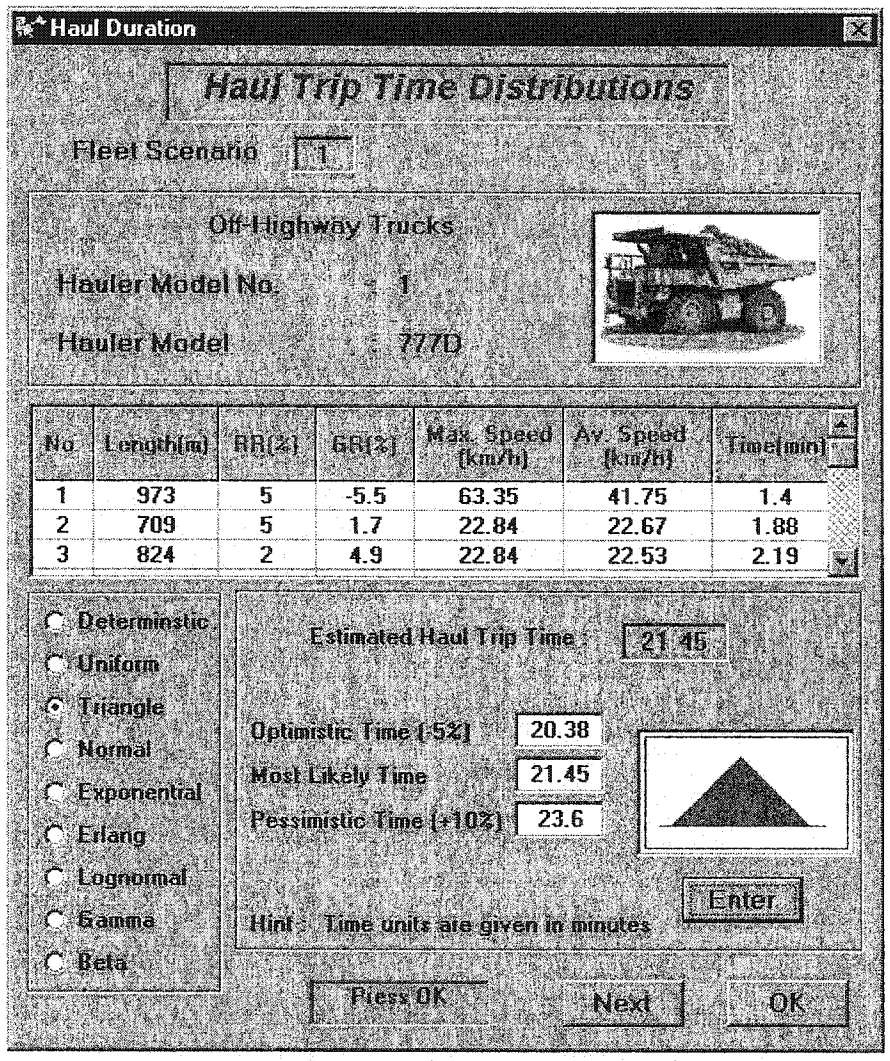


Figure 7.16 Haul Activity Distribution User Interface

7.2.6 Run Simulation

Simulation analysis is performed in the simulation engine (*EMSP*) of *SimEarth*. *EMSP* is run under the *SimEarth*'s environment. Simulation parameters are set through *SimEarth*'s interactive user interfaces (see Figure 7.17). Those parameters include: 1) condition(s) of termination for the simulation analysis (total scope of work or simulation time), 2) interaction among equipment in the fleet under consideration, and 3) storage of simulation data for later use in animation.

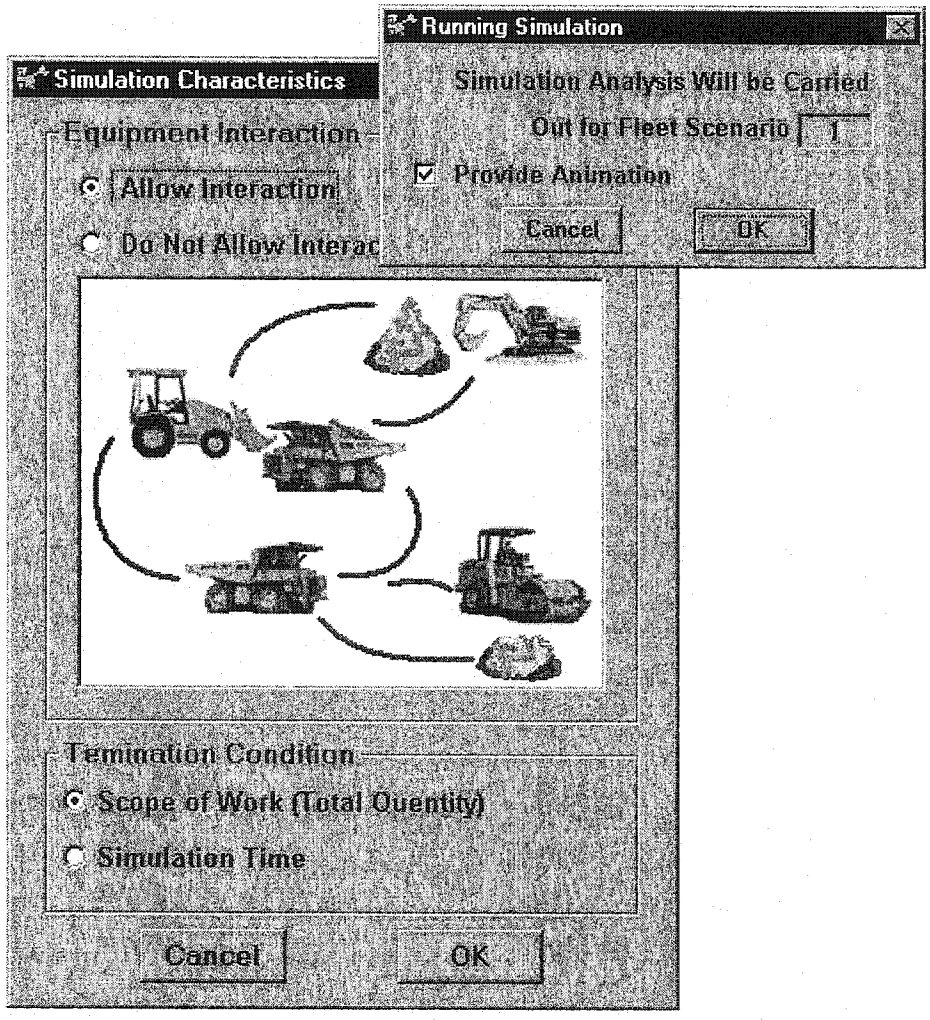


Figure 7.17 Simulation Parameters User Interface

7.3 Output Reporting Module

7.3.1 Static Sub-Module

The static sub-module extracts data from all of the system's components including: equipment cost application (*ECA*); equipment database application (*EDA*); earthmoving markup application (*EMMA*); earthmoving genetic algorithm (*EM_GA*); and earthmoving simulation program (*EMSP*). A set of user interfaces has been developed to facilitate the use of *ORM* static sub-module. VBA is used in MS Excel 97 environment to automate the process of generating reports (see Appendix B). Figure 7.18 depicts the initial and main menus for the static sub-module, which generates *EMSP* output.

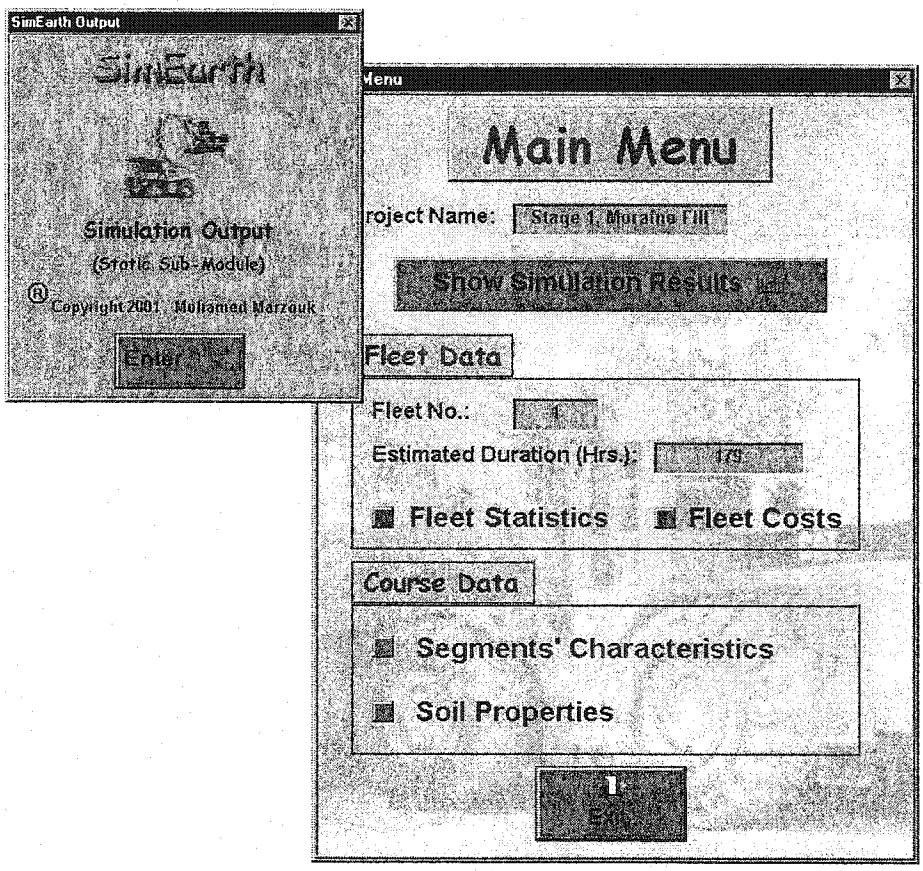


Figure 7.18 ORM Main Menu for EMSP Outputs (Static Sub-Module)

Different types of reports are obtained from *EMSP* (see Appendix B). These are: 1) simulation results, 2) fleet statistics, 3) fleet cost, 4) segments' characteristics and 5) soil parameters. These reports are generated in an interactive tabular and graphical format as depicted in Figure 7.19.

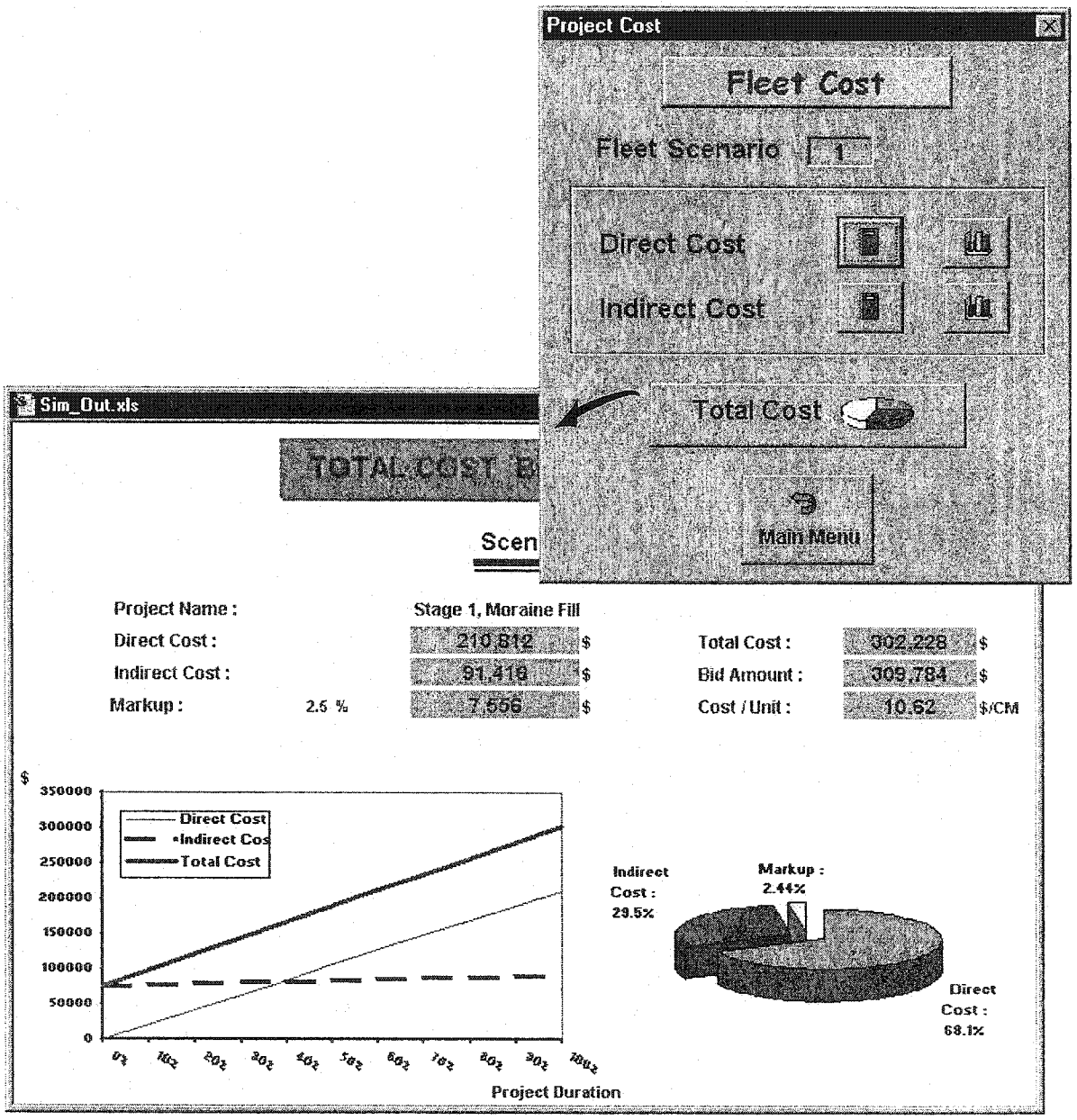


Figure 7.19 Static Sub-Module Interactive User Interfaces

7.3.2 Dynamic Sub-Module

The dynamic sub-module supports animation to represent the overall system dynamics, the interaction between customer and server resources, the bottlenecks of the system, and the delay associated with the individual resources. This sub-module utilizes Proof animation software (2000) which extracts animation data from an external file, prepared by the `Trace_Animation` class of the developed *EMSP*. Appendix A includes a part of the `SimEarth.atf` file, used for animation. Figure 7.20 illustrates a snap shot for a layout of an earthmoving operation at 59.23 simulation time. Figures 7.21 and 7.22 depict animated screens with a closer look at load and dump zones, respectively.

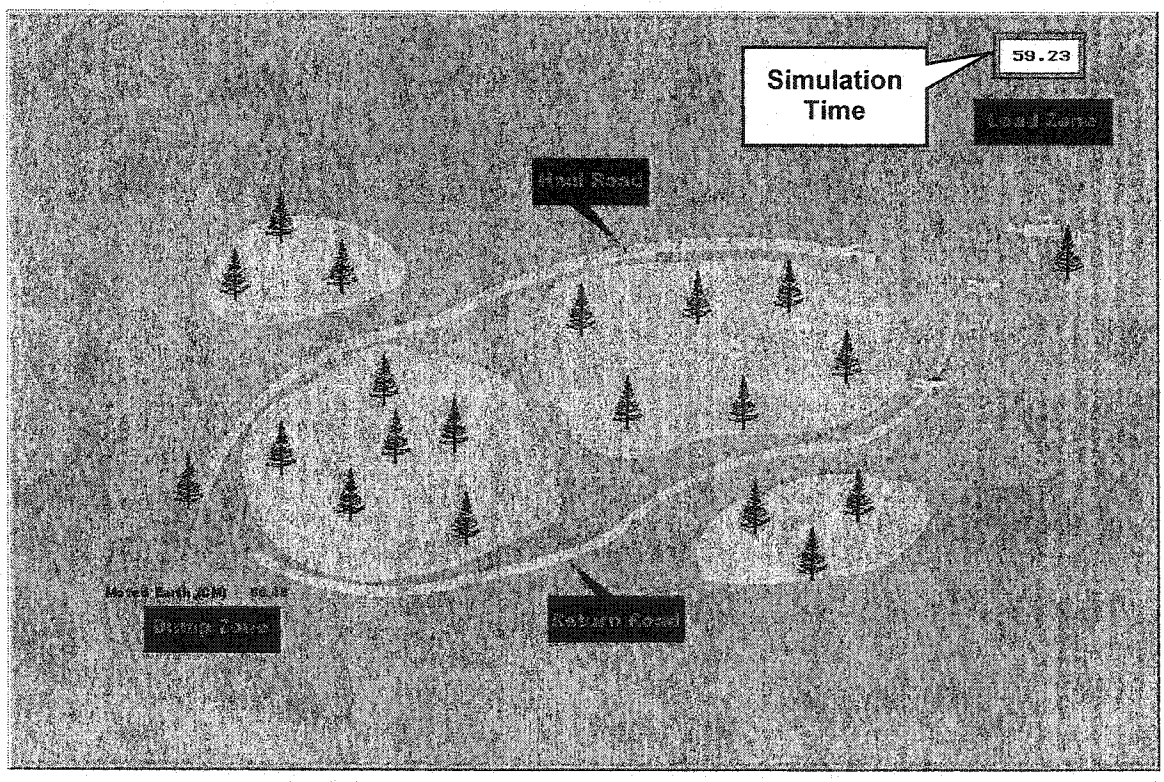


Figure 7.20 Dynamic Sub-Module Animated Layout

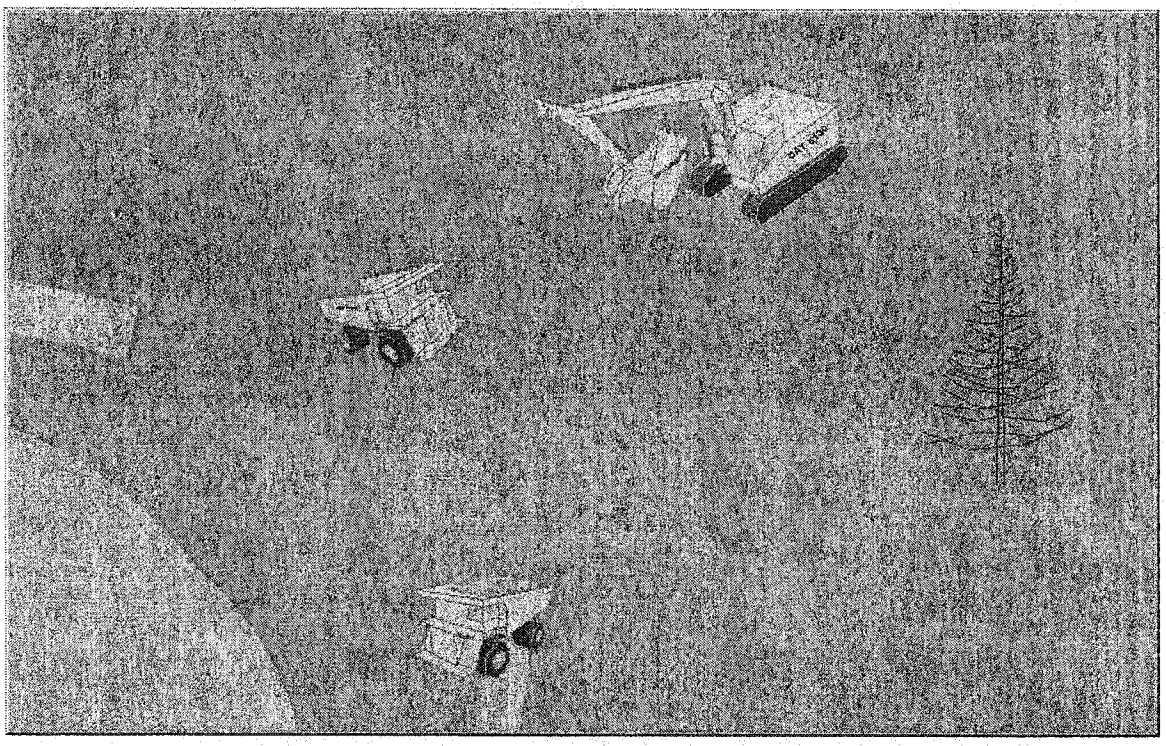


Figure 7.21 Animated Screen at Load Zone

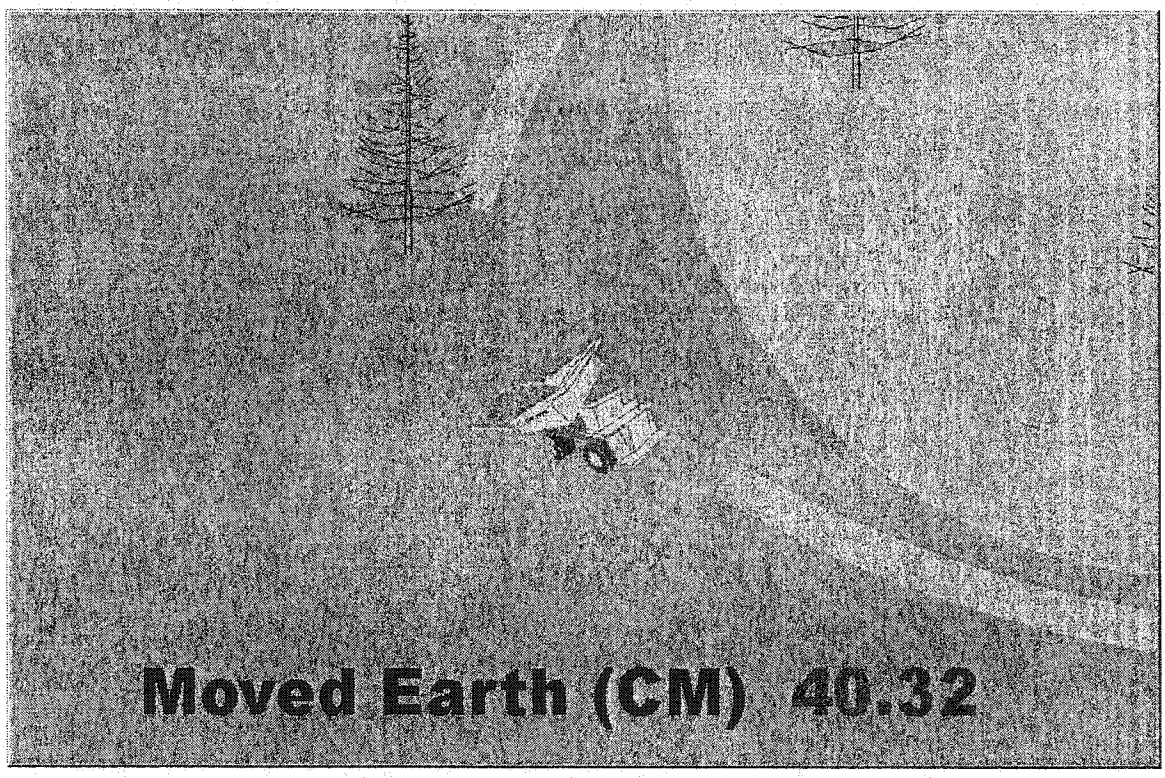


Figure 7.22 Animated Screen at Dump Zone

It should be noted that geometry of equipment was drawn using AutoCAD software. Subsequently, the geometry was imported and defined to dynamic sub-module. Table 7.2 describes example of commands written in the `SimEarth.atf` file. These commands were prepared by the `Trace_Animation` class as input to Proof Animation software.

Table 7.2 Dynamic Sub-Module Commands

Command	Description
`Create LOADER 1`	Create object of loader type that has identification (ID) equals to 1.
`Create TRUCK_H 11`	Create object of haul truck type that has identification (ID) equals to 11.
`Create TRUCK_R 11`	Create object of return truck type that has identification (ID) equals to 11.
`Create TRUCK_D 11`	Create object of dump truck type that has identification (ID) equals to 11.
`Place 1 On LoaderQueue at end`	Place object (ID=1) at the end of a path (begging of a queue) called `LoaderQueue`.
`Place 11 440 204`	Place object (ID=11) at a particular position (x=440 and Y=204)
`Destroy 11`	Destroy object (ID=11).
`Time 3.41264`	Set simulation time at 3.41264
`Place 11 On HaulRoad`	Place object (ID=11) on a path (HaulRoad)
`set 11 travel 24.9718`	Object (ID=11) will spent 24.9718 simulation time units travelling across its current pass.
`write Quantity 504`	Update the quantity of moved earth message to be 504.
`End`	End animation

7.4 Case Study

7.4.1 Case Description

This case study considers the construction of Saint-Margurerite-3 (SM-3) dam. The dam is the highest rockfill dam in the province of Quebec, located on "Saint-Marguerite" river, 700 km northeast of Montreal (see Figure 7.23). Figure 7.24 shows the SM-3, the rockfill dam after construction. SM-3 is a $2 billion project, developed to generate 2.7 terawatthours (TWh) of electricity annually. The project consists of four main components (Hydro-Quebec 1999): 1) a rockfill dam, 2) an 882-megawatt (MW) power station, 3) a headrace tunnel to direct water to the power station and 4) a spillway to discharge excess water from the reservoir. The dam location was chosen to benefit from a 330 m water head, seven times the height of Niagara Falls.

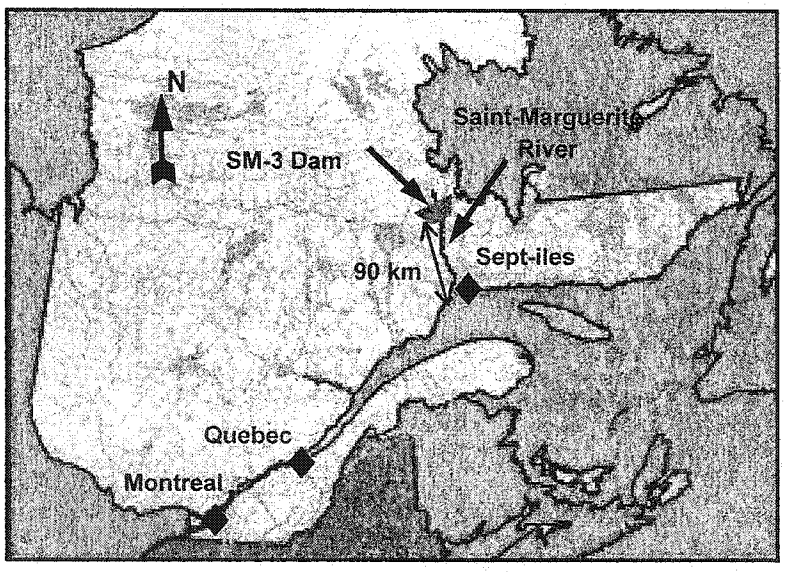


Figure 7.23 Dam Location across Saint-Marguerite River

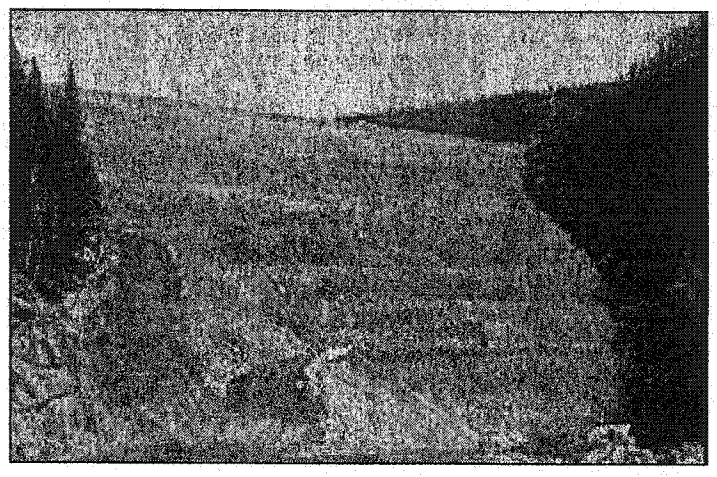


Figure 7.24 SM-3 Dam after Construction (EBC 2001)

A temporary diversion tunnel was excavated and a rockfill cofferdam was constructed to dry the dam site during construction (see Figure 7.25).

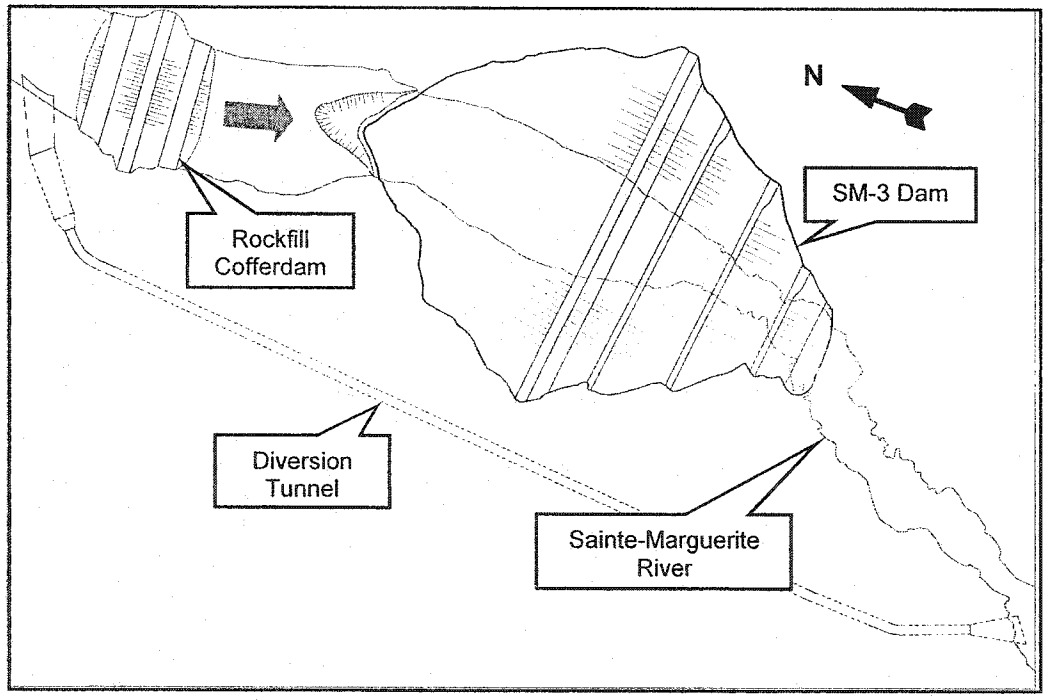


Figure 7.25 Temporary Diversion Tunnel and Rockfill Dam

In view of the cold weather conditions at the project site, construction can only be performed between the beginning of April till the end of November. This study focuses only on the rockfill dam with an objective of minimizing total project duration. The data used in this case (see Table 7.3) was retrieved from publications (Hydro-Quebec 1999, Peer 1998, Hydro-Quebec 2001, Peer 2001) and extracted directly from project drawings.

Table 7.3 Data of the Dam (Hydro-Quebec 2001)

Height :	**171 m**
Length at crest :	**378 m**
Crest width :	**10 m**
Base width :	**500 m**
Crest Elevation :	**410 m**
Max. normal water level :	**407 m**
Min. normal water level :	**393 m**
Volume of fill :	**6.3 million m^3**

7.4.2 Case Modeling

The owner targeted the completion of the dam construction in three years. To model the rockfill dam, the following were considered: 1) the quantities and type of soils used to fill the body of the dam (i.e. scope of work), 2) the locations of soils' borrow bits, 3) the target three-year construction duration, 4) the constraints paused by the relatively short construction season, 5) the equipment used to perform the work, and 6) the indirect cost components. The dam consists of several soil types with different size and compaction requirements. For simplicity, three soil types were considered in the modeling of the dam: 1) compacted

moraine (clay), 2) granular (sand and gravel) and 3) rock. The total volume of the soil considered in the modeling was 6.3 million m^3 (Hydro-Quebec 1999). The actual excavated natural soil from the riverbed was estimated to be 1,038,000 m^3 (Peer 2001). In view of the relatively short construction season and the targeted project duration, the project was phased in three stages, each spanning a construction season as shown in Figure 7.26. The type and size of the different soils used to fill the body of the dam are shown in Figure 7.27

Table 7.4 summarizes the scope of work in each stage (estimated from the project drawings). The properties of soils used to fill the body of the dam and the clay excavated from the riverbed are given in Table 7.5. These soils were borrowed from three borrow pits during construction.

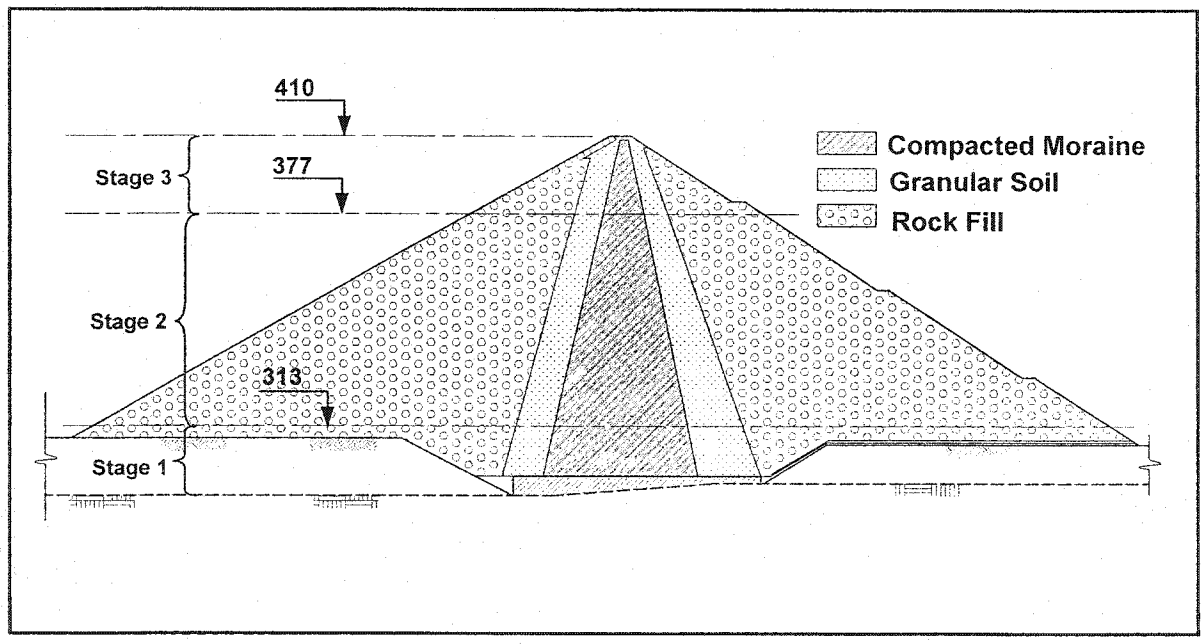


Figure 7.26 Typical Cross Section of the Dam

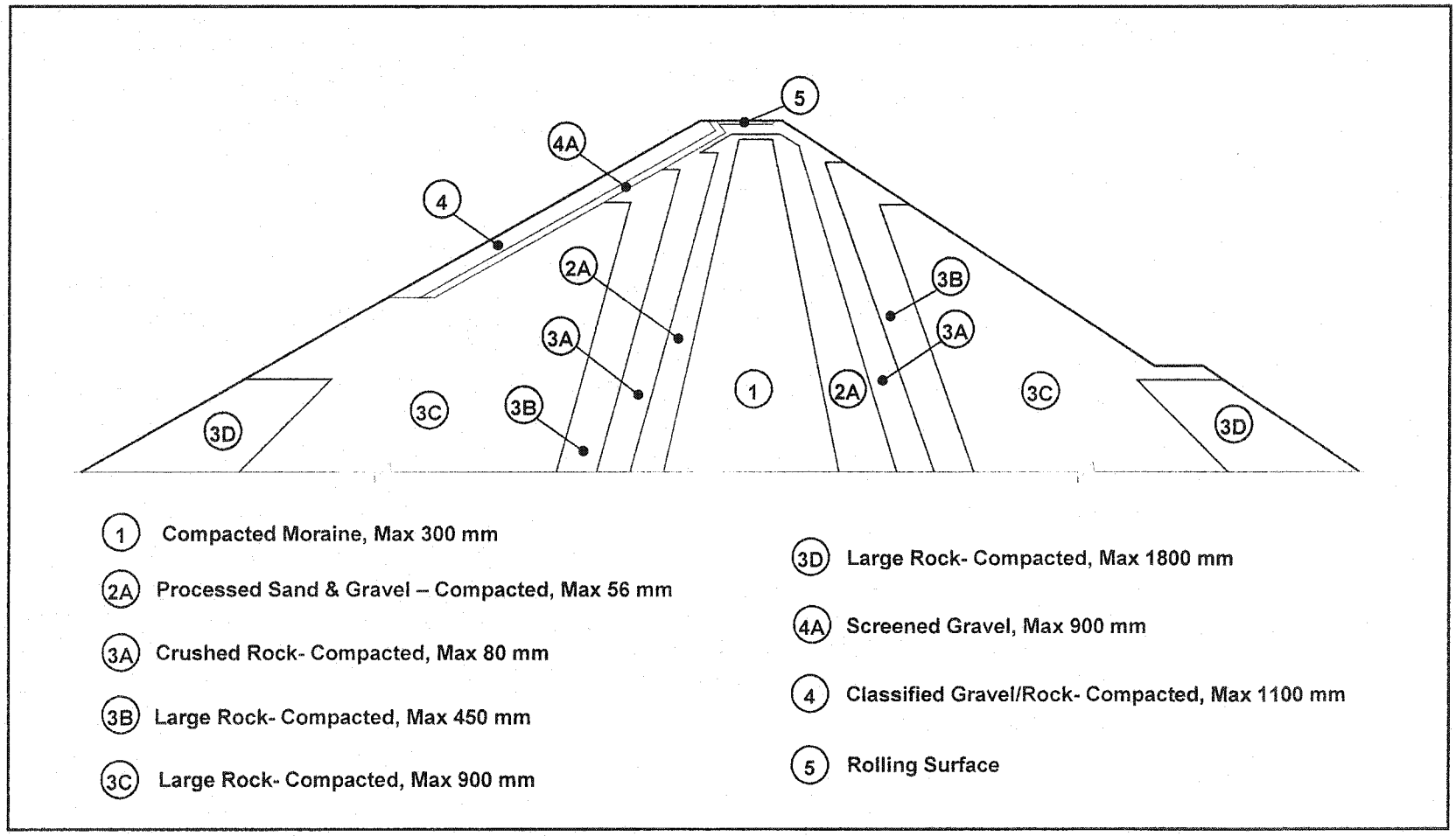


Figure 7.27 Characteristics of Soils used in the Dam

Table 7.4 Scope of Work for Earth Fill (Bank m^3)

Soil Type	Stage 1 Elev. (313 m)	Stage 2 Elev. (377 m)	Stage 3 Elev. (410 m)	
Moraine	29,200	555,900	269,900	855,000
Granular	14,500	286,500	139,000	440,000
Rock	192,700	3,209,400	1,602,900	5,005,000
Total	236,400	4,051,800	2,011,800	6,300,000

Table 7.5 Soil Properties

Soil Type	Loose Density (t/m^3)	Bank Density (t/m^3)	Load Factor (%)
Moraine	1.66	2.02	100
Granular	1.72	1.93	90
Rock	1.66	2.73	80
Excavated Clay	1.6	2.4	100

The characteristics of haul roads (i.e. lengths, number of segments per road and the grade of each segment) were determined from the contour drawings, which establish the profiles of the proposed haul roads as shown in Figure 7.28. Figure 7.29 depicts the borrow pits and the dumping zone locations relative to the dam. The data pertaining to the characteristics of the different haul roads are listed in Tables 7.6, 7.7, 7.8, and 7.9, receptively. The scope of work and the targeted project duration clearly influence the amount of equipment involved in each fleet scenario and the number of shifts in each stage. Due to the large amount of earth to be moved in the second stage (4,051,800 m^3), two 8-hours shifts were considered in the study (Peer 2001).

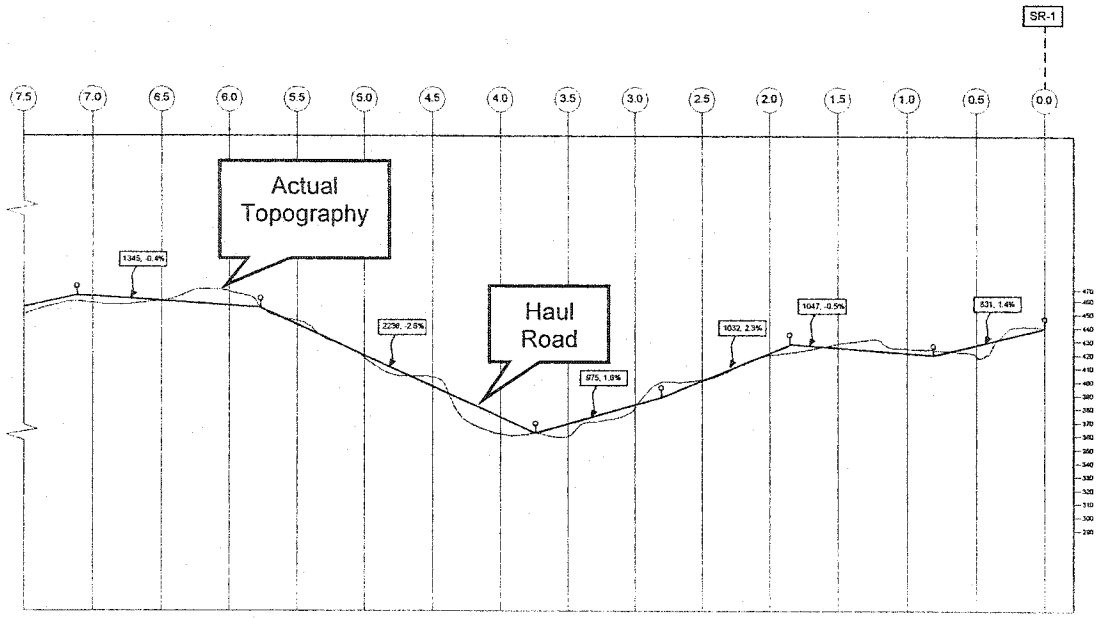
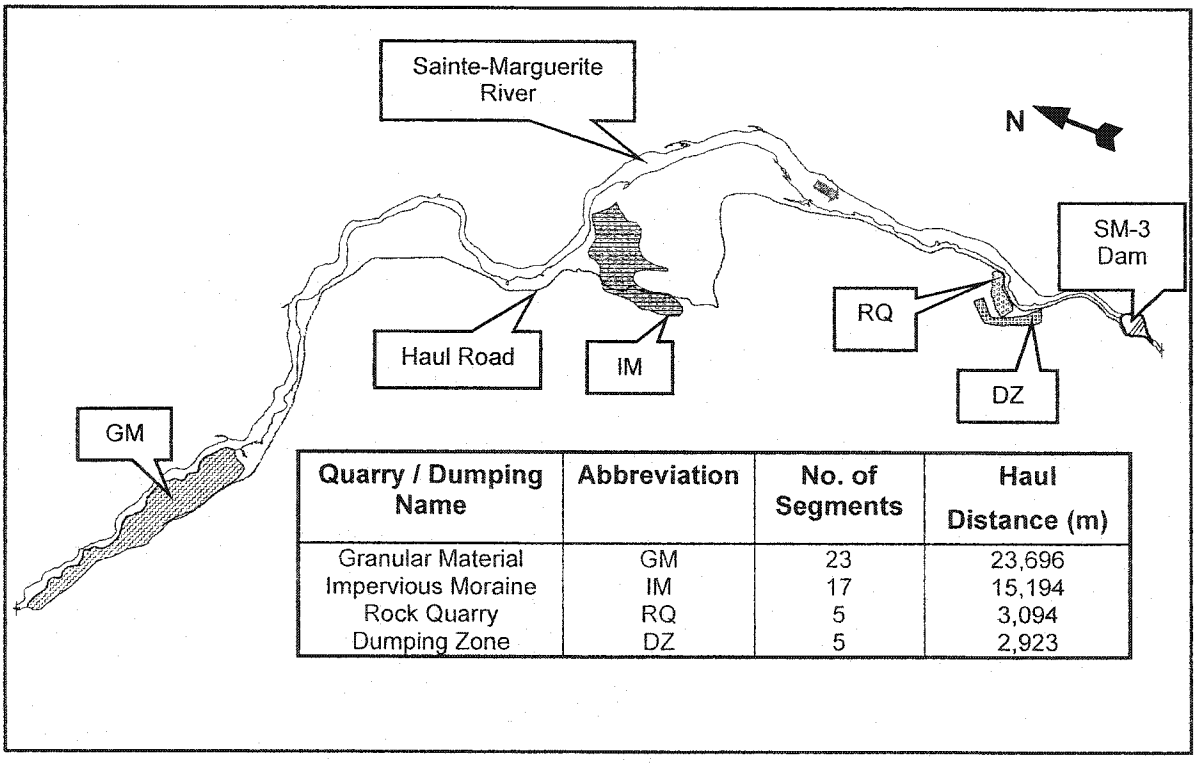


Figure 7.28 An Example of Cross-Sectional Profile for Haul-Roads



Quarry / Dumping Name	Abbreviation	No. of Segments	Haul Distance (m)
Granular Material	GM	23	23,696
Impervious Moraine	IM	17	15,194
Rock Quarry	RQ	5	3,094
Dumping Zone	DZ	5	2,923

Figure 7.29 Quarry and Dumping Zones

Table 7.6 Haul Road from GM to the Dam

Seg. No.	Length (m)	% Grade	% RR	% TR
1	230	2.5	4.0	6.5
2	315	2.4	2.0	4.4
3	2221	-0.5	2.0	1.5
4	950	0.8	2.0	2.8
5	1,062	-0.6	2.0	1.4
6	1,094	-0.5	2.0	1.5
7	2,911	-0.1	2.0	1.9
8	1,310	0.2	2.0	2.2
9	915	4.6	2.0	6.6
10	1,167	1.9	2.0	3.9
11	899	0.7	2.0	2.7
12	1,023	-0.5	2.0	1.5
13	1,415	-5.9	2.0	-3.9
14	891	-0.5	2.0	1.5
15	962	0.4	2.0	2.4
16	708	-0.2	2.0	1.8
17	949	-0.6	2.0	1.4
18	1,031	1.4	2.0	3.4
19	1,006	-0.6	2.0	1.4
20	787	0.1	2.0	2.1
21	710	-0.2	2.0	1.8
22	955	3.3	2.0	5.3
23	185	0	2.0	2.0
	Σ 23,696			

Table 7.7 Haul Road from the Dam to DZ

Seg. No.	Length (m)	% Grade	% RR	% TR
1	185	0	2.0	2.0
2	899	2.4	2.0	4.4
3	710	0.2	2.0	2.2
4	287	-0.1	2.0	1.9
5	842	4.1	4.0	8.1
	Σ 2,923			

Table 7.8 Haul Road from IM to the Dam

Seg. No.	Length (m)	% Grade	% RR	% TR
1	973	-5.5	5.0	-0.5
2	709	1.7	5.0	6.7
3	824	4.9	2.0	6.9
4	1,167	1.9	2.0	3.9
5	899	0.7	2.0	2.7
6	1,023	-0.5	2.0	1.5
7	1,415	-5.9	2.0	-3.9
8	891	-0.5	2.0	1.5
9	962	0.4	2.0	2.4
10	708	-0.2	2.0	1.8
11	949	-0.6	2.0	1.4
12	1,031	1.4	2.0	3.4
13	1,006	-0.6	2.0	1.4
14	787	0.1	2.0	2.1
15	710	-0.2	2.0	1.8
16	955	3.3	2.0	5.3
17	185	0	2.0	2.0
	Σ 15,194			

Table 7.9 Haul Road from RQ to the Dam

Seg. No.	Length (m)	% Grade	% RR	% TR
1	457	-2.8	4.0	1.2
2	787	0.1	2.0	2.1
3	710	-0.2	2.0	1.8
4	955	3.3	2.0	5.3
5	185	0	2.0	2.0
	Σ 3,094			

The recommendations of equipment manufacturers (Caterpillar Performance Handbook 1997) were followed to ensure compatibility between haulers and loaders. These recommendations include for each hauler its suitable loader, along with the number of loader passes to fill the hauler. A triangular probability distribution was selected for the duration of the load, haul, return, spread and compact activities and a uniform distribution for duration of dump activity (see Tables 7.10 and 7.11). The assumptions made here, for these distributions, do not pose limitations on the developed system. The user can similarly select the most appropriate distribution imbedded in the system. The job efficiency factor was assumed to be 78%, which represents *Good* job conditions and *Excellent* management conditions (Nunnally 1998). Travel time for haul and return activities was estimated using *HTTA* and data pertaining to haul segments and haulers load percent (Column 7, Table 7.10).

Based on haulers' cycle time (summation of Columns 8, 9, 10 and 11 in Table 7.10), the quantity of soil in the stage under study, and the hauled earth per trip, the required duration based on the use of a single hauler was obtained. Subsequently, the number of haulers was calculated, respecting the project time constraints. For the *F1_Mor* fleet, for example, the approximated cycle time is 44 minutes, the quantity of moraine in the first stage is 29,182 m^3 and the hauled earth per trip is 81.67 ton. Accordingly, the required duration, based on the use of a single hauler, can be calculated using the following formula:

Table 7.10 Configuration of Fleets and their Time Distributions

Fleet Name	Hauler Model	Loader Model	Bucket Cap. (m^3)	No. of Passes	Hauled Earth (ton)	%Load	Load Activity Dist.	Haul Activity Dist.	Dump Activity Dist.	Return Activity Dist.
F1_Mor	777D	992G	12.3	4	81.67	84.2	T(2.43, 2.56, 2.82)	T(20.38, 21.45, 23.6)	U(1.9, 2.2)	T(17.26, 18.17, 19.99)
F2_Mor	773D	990SII	9.2	3	45.82	87.6	T(1.82, 1.92, 2.11)	T(19.47, 20.5, 22.55)	U(1.6, 1.9)	T(16.71, 17.59, 19.35)
F3_Mor	769C	988F	6.9	3	34.36	93.3	T(1.82, 1.92, 2.11)	T(19.1, 20.11, 22.12)	U(1.4, 1.6)	T(16.76, 17.64, 19.4)
F1_Gran	777D	992G	12.3	5	95.2	98.1	T(3.04, 3.2, 3.52)	T(29.75, 31.32, 34.45)	U(1.9, 2.2)	T(22.75, 23.95, 26.35)
F2_Gran	773D	990SII	9.2	3	42.72	82	T(1.82, 1.92, 2.11)	T(26.82, 28.23, 31.05)	U(1.6, 1.8)	T(22.07, 23.23, 25.55)
F3_Gran	D400D	988F	6.9	3	31.67	87.2	T(1.82, 1.92, 2.11)	T(30.72, 32.34, 35.57)	U(1.3, 1.5)	T(25.18, 26.51, 29.16)
F1_Rock	777D	992G	12.3	5	81.67	84.2	T(3.94, 4.57, 4.15)	T(4.3, 4.53, 4.98)	U(1.9, 2.2)	T(3.17, 3.34, 3.67)
F2_Rock	773D	990SII	9.2	4	48.87	93	T(3.15, 3.32, 3.65)	T(4.17, 4.39, 4.83)	U(1.7, 1.9)	T(3.1, 3.26, 3.59)
F3_Rock	D400D	988F	6.3	4	33.46	92.2	T(3.15, 3.32, 3.65)	T(4.59, 4.83, 5.31)	U(1.4, 1.6)	T(3.45, 3.73, 4.1)
F1_Ex_C	777D	5130ME	10.5	5	84	86.6	T(3.04, 3.2, 3.52)	T(5.34, 5.62, 6.18)	U(1.9, 2.2)	T(2.93, 3.08, 3.39)
F2_Ex_C	773D	375L	4.59	7	51.41	98.3	T(4.26, 4.48, 4.93)	T(5.32, 5.6, 6.16)	U(1.6, 1.9)	T(2.86, 3.01, 3.31)

Note:

U (N1, N2); U : Uniform Distribution, N1: Lower Limit and N2: Upper Limit

T (N1, N2, N3); T : Triangle Distribution, N1: Lower Limit, N2: Mode and N3: Upper Limit

Table 7.11 Spread and Compact Time Distributions

Activity	Equipment	Productivity (m^3/Cycle)	Distribution
Spread	D8R	27	T(2.47, 2.6, 2.86)
Compact	CS-583C	19.1	T(1.8, 1.9, 2.09)

$$\text{Dur. (hrs.)} = \frac{\text{Approx. Cycle time (mins.)}}{60} \quad \frac{\text{Scope of Work } (m^3)}{\text{Cycle Production } (m^3)} \qquad (7.3)$$

$$\text{Dur.}_{F1\_Mor\,(777D)} = \frac{44}{60} * \frac{29{,}200 * 2.02}{81.67} \cong 530 \text{ hrs.}$$

$$\text{Dur.}_{F1\_Mor\,(992G)} = \frac{2.56}{60} * \frac{29{,}200}{12.3} \cong 102 \text{ hrs.}$$

Assigning three haulers of 777D model will finish the task of moving moraine (which represents 12% of the first stage fill) in approximately 178 hrs (one month) which is greater than the required duration utilizing one loader of 992G model. The number of spreaders and compactors is calculated based on their productivity rates. Productivity rates for spreaders were calculated utilizing EMF software (1998) for different models of dozers. Productivity rates for compactors were obtained directly from the Caterpillar Performance Handbook (1997). These productivity rates are stored in the developed dialog boxes for each piece of equipment to facilitate its use in future applications. Similarly, the thirty-one fleet configurations shown in Table 7.12 (Columns 6, 7 and 8) were generated. It should be noted that each fleet has been represented by a one-dimensional array depicting the number of each piece of equipment used. For example, fleet (3,1,1,1), used in F1_Mor, consists of three haulers of model 777D, one loader of model 992G, one spreader of model D8R and one compactor of model CS-583C.

Indirect cost was modeled utilizing the cost components shown in Table 7.13. It should be noted that the time-independent portion of the field support cost represents actual figures. The time-related portion of the field support cost (i.e. plant operation), field general and administrative, and field summary were not

disclosed for confidentiality reasons. They are assumed, in this study, to be $250,000 and $500,000 for the one 8-hours shift and the two 8-hours shifts, respectively.

Table 7.12 Number of Configured Equipment in three Project Stages

Fleet Name	Hauler Model	Loader Model	Spreader Model	Compactor Model	Stage 1	Stage 2	Stage 3
F1_Mor	777D	992G	D8R	CS-583C	(3, 1, 1, 1)	(12, 2, 2, 2)	(5, 1, 1, 1)
F2_Mor	773D	990SII	D8R	CS-583C	(4, 1, 1, 1)	(14, 2, 2, 2)	(6, 1, 1, 1)
F3_Mor	769C	988F	D8R	CS-583C	(6, 1, 1, 1)	(17, 3, 2, 2)	(10, 1, 1, 1)
F1_Gran	777D	992G	D8R	CS-583C	(3, 1, 1, 1)	(8, 1, 1, 1)	(4, 1, 1, 1)
F2_Gran	773D	990SII	D8R	CS-583C	(4, 1, 1, 1)	(9, 1, 1, 1)	(5, 1, 1, 1)
F3_Gran	D400D	988F	D8R	CS-583C	(6, 1, 1, 1)	(13, 1, 1, 1)	(8, 1, 1, 1)
F1_Rock	777D	992G	D8R	CS-583C	(4, 1, 1, 1)	(35, 5, 6, 6)	(20, 3, 3, 3)
F2_Rock	773D	990SII	D8R	CS-583C	(5, 1, 1, 1)	(40, 5, 6, 6)	(25, 3, 3, 3)
F3_Rock	D400D	988F	D8R	CS-583C	(8, 1, 1, 1)	(45, 6, 6, 6)	(30, 4, 3, 3)
F1_Ex_C	777D	5130ME	D8R		(20, 3, 3 , 0)		
F2_Ex_C	773D	375L	D8R		(24, 4, 3 , 0)		

Note:

(N1, N2, N3, N4); N1: No. of haulers, N2: No. of loaders, N3: No. of spreaders and N4: No. of compactors

Table 7.13 Components of Indirect Cost

Category/Sub Category	Payment Type	Component
Field support cost	Time-Independent Time-Related *	46,100,000 150,000
Field general and administrative	Time-Related *	100,000
Field summary cost	Time-Independent *	5% of Direct cost

** Assumed Components*

7.4.3 Analysis of the Case

Upon entering the input data to *SimEarth* and requesting the fleet that provides minimum total duration, *SimEarth* first triggers its simulation engine (*EMSP*) to perform pilot runs (see Figure 7.30). Subsequently, simulation analysis is performed for each of the recommended fleets by specifying the number of simulation runs and activating the "*Analyze*" function (see Figure 7.30).

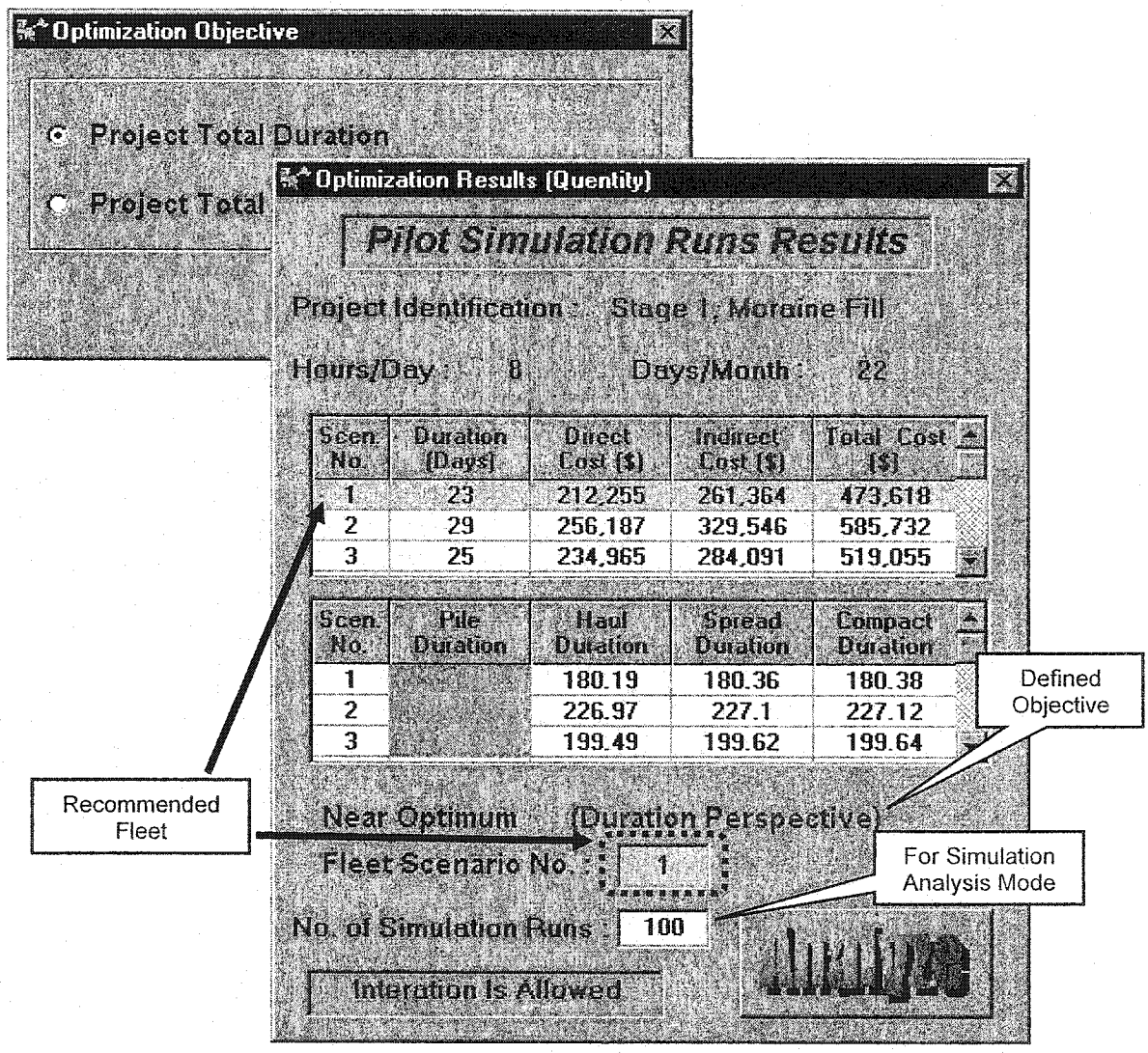


Figure 7.30 Results of Pilot Simulation

Different reports can then be generated in a graphical form (see Figure 7.31) and/or a tabular form as shown in Figure 7.32. Tables 7.14 and 7.15 list the estimated durations obtained from the simulation analysis and their associated direct cost. Table 7.16 provides the total cost, direct and indirect, of the project.

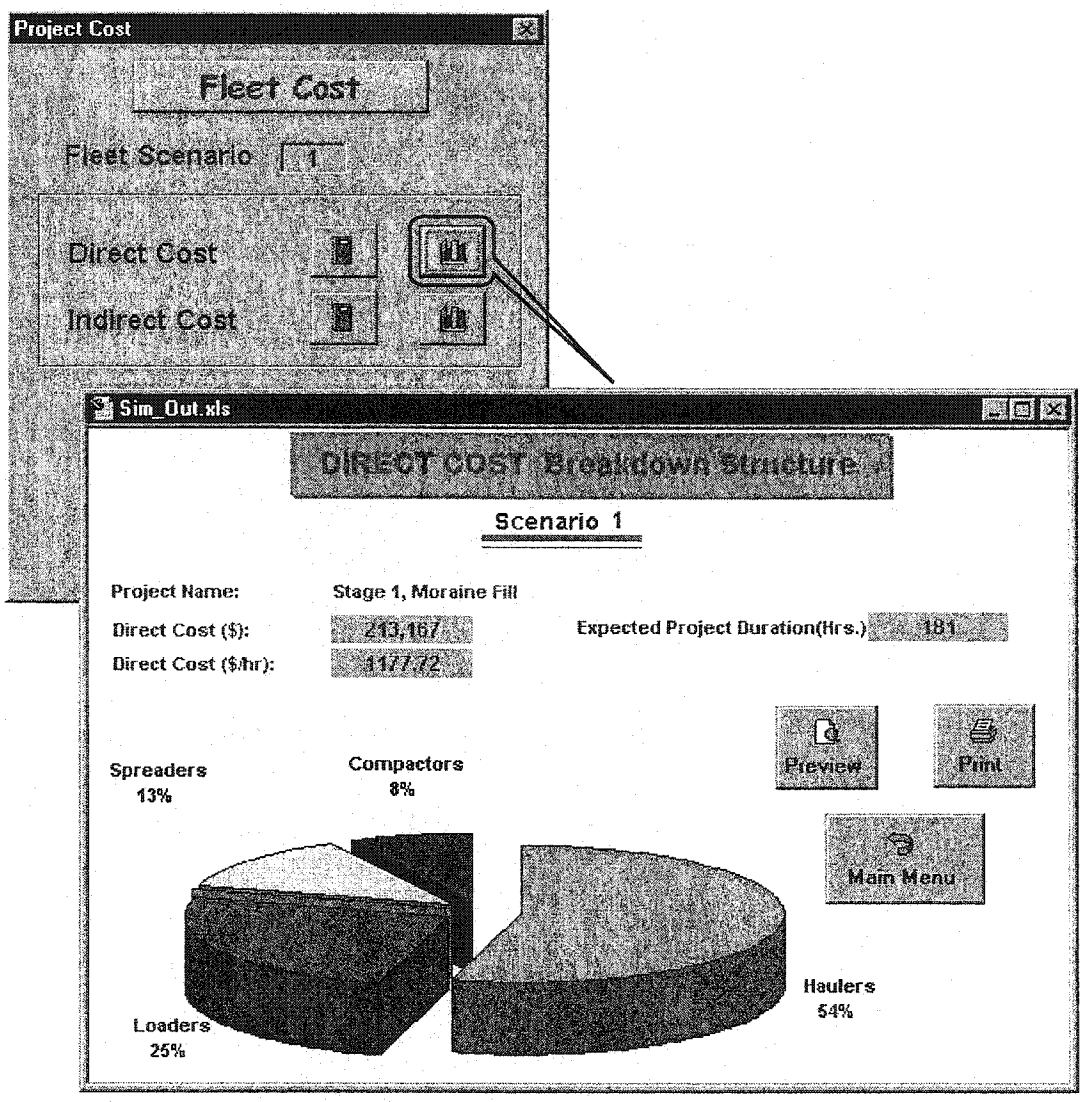


Figure 7.31 Graphical Report (F1_Mor, Stage 1)

Table 7.14 Estimated Durations (Hours)

	Stage 1			Stage 2			Stage 3		
Fleet Name	Main Activities	Spread Activity	Compact Activity	Main Activities	Spread Activity	Compact Activity	Main Activities	Spread Activity	Compact Activity
F1_Mor	181	181	181*	857	857	857*	1000	1000	1000*
F1_Gran	101	101	101*	746	746	746*	722	722	722*
F1_Rock	452	516	532*	1505	1505	1505*	1253	1430	1476*
F1_Ex_C	458*	812							
Total Duration	1272 hrs. = 159 Days = 7.2 Months			3108 hrs. = 195 Days = 7.5 Months			3198 hrs. = 200 Days = 7.7 Months		

** Last Activity*

Table 7.15 Estimated Direct Cost (Dollars)

Fleet Name	Stage 1	Stage 2	Stage 3
F1_Mor	213,167	3,113,601	1,603,620
F1_Gran	118,950	1,672,883	1,004,064
F1_Rock	645,577	15,638,049	8,837,741
F1_Ex_C	3,117,007		
Total Direct Cost	4,094,701	20,424,532	11,445,424

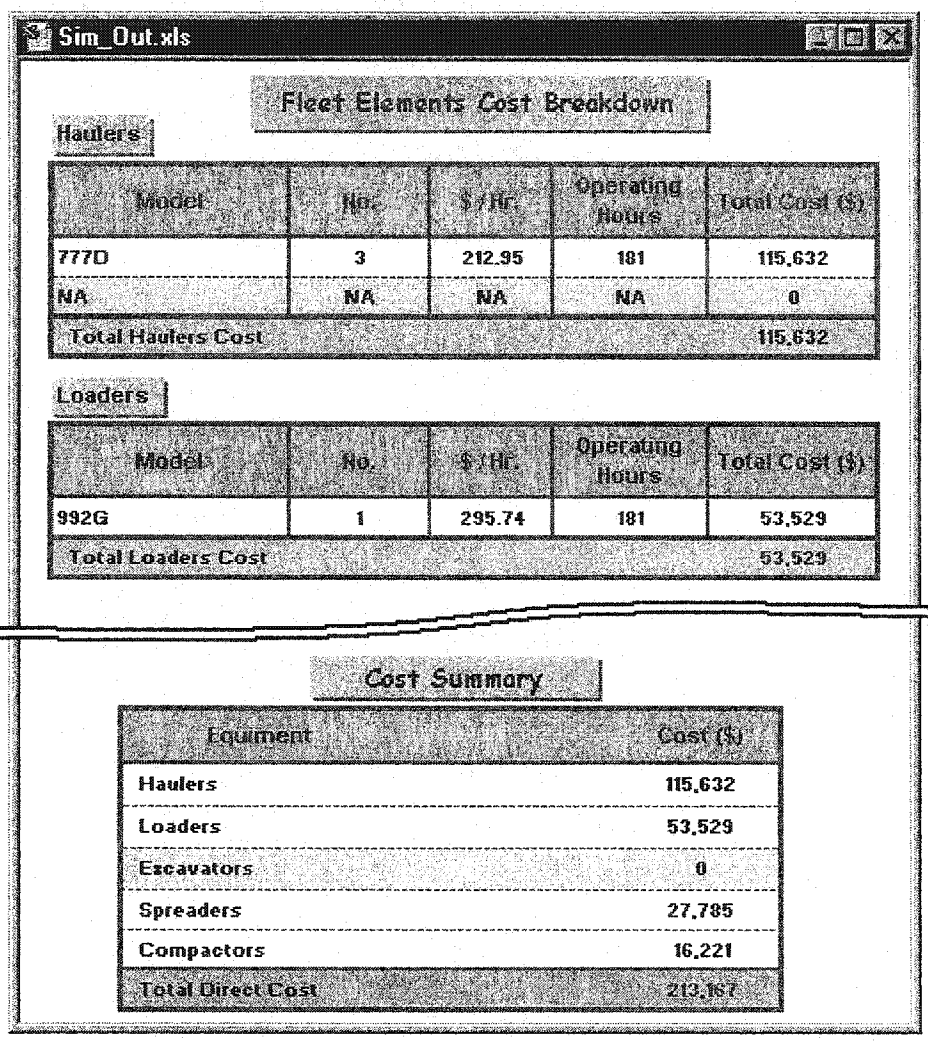


Figure 7.32 Tabular Reports (F1_Mor, Stage 1)

Table 7.16 Project Total Cost

		Indirect Cost	
Fleet Name	Direct Cost	Time Related	Lump sum
Stage 1	4,094,701	1,800,000	46,100,000
Stage 2	20,424,532	3,750,000	+ 0.05*35,964,657
Stage 3	11,445,424	3,850,000	+ 6,075,000
	35,964,657	63,373,233	
Markup (2.5%)	2,331,572		
Total Cost	**101,669,462**		

Cost of access roads

Table 7.14 indicates that the longest duration (7.7 months, Stage 3) meets project time constraint, accounting for the duration of the construction season. It should be noted that the study has been carried out from an owner's perspective to provide a preliminary estimate of the project's total cost. It should be noted that, in the original project, an additional $90,000,000 was assigned for construction of roads/highways to serve the overall project. The amount includes the cost of high-quality roads to make the site and worker's camp accessible over a total length of 100 km. Due to the lakes and rivers in the area, 350 culverts have been constructed in addition to a bridge that crosses Saint-Marguerite river (180 m) near the power station (Hydro-Quebec 1999). In the modeling of the case, a portion of that cost ($6,075,000) was assigned to the dam's indirect cost component (see Table 7.15). This portion is assumed to be proportional to the dam's actual contract value $135 million (Peer 1998) in relation to the total cost of the project ($2 billion).

7.5 Summary

This chapter presented the implementation of developed system *SimEarth*, dedicated for earthmoving operations. The interactive screens of the *SimEarth* have been coded using Visual Basic 6.0 to facilitate data entry process. The developed system is a window application that runs on Microsoft Windows 95, 98, 2000 or NT. The different user interfaces have been illustrated along with their required data. These interfaces are dedicated for entering: 1) soil quantities and characteristics; 2) haul road characteristics; 3) fleet configurations; 4) activity

durations; and 5) running simulation. The modeling of the indirect cost components, considered in *SimEarth*, and their categories have been outlined.

The chapter also presented the generated reports from the system's output reporting module (*ORM*) which are: 1) tabular and chart forms (generated from static sub-module) and 2) animation form (generated from dynamic sub-module). A comprehensive case study of an actual project was also presented in order to illustrate the process of fleet selection using the developed system and demonstrate its features. The project involves the construction of a large rockfill dam, located in the northern part of the province of Quebec.

CHAPTER 8

SUMMARY AND CONCLUSIONS

8.1 Summary

This thesis presented a methodology for optimizing earthmoving operations using computer simulation and genetic algorithms. The research aims at the development of an optimization tool, geared towards the selection of a near optimum fleet utilizing computer simulation. The selection of equipment fleet is performed taking into account the availability of equipment. The developed methodology has been implemented in a prototype software system named (*SimEarth*) which consists of five main components: 1) EarthMoving Simulation Program (*EMSP*); 2) Equipment Cost Application (*ECA*); 3) Equipment Database Application (*EDA*); 4) EarthMoving Genetic Algorithm (*EM_GA*) and 5) Output Reporting Module (*ORM*). Beside these main components, *SimEarth* is supported by: a) Hauler's Travel Time Application (*HTTA*) and b) EarthMoving Markup Application (*EMMA*). All these components are integrated and dynamically lined within *SimEarth* environment which has been coded utilizing Visual Basic 6.0.

EMSP represents the main component of *SimEarth*. It has been designed utilizing discrete event simulation (DEVS) technique, and employing object-

oriented concepts. Three-phase simulation, rather than process interaction, was used to control the system dynamics by tracking the activities of the simulated process (e.g., piling, load, haul, dump, spreading, and compacting). The three-phase simulation approach is considered most appropriate for object oriented simulation (OOS) especially when there are many entities involved in the process being modeled (Pidd 1995). Object oriented simulation has been utilized by defining the simulated process components into objects that interact and communicate together. Different features of object orientation have been utilized including objects, classes, encapsulation, dynamic data structure, and polymorphism. *EMSP* has been coded utilizing Visual C++ 6.0. A numerical example of an actual case has been worked out to validate the developed *EMSP* and demonstrate its capabilities.

ECA has been developed for estimating equipment cost, accounting for the three main categories of cost (owning costs, operating costs, and wages of equipment operators). It is essentially a spreadsheet application, designed to provide the user with the total hourly owning and operating costs along with their respective breakdown. In order to estimate owning costs, the calculations are carried out in a highly interactive user-friendly environment supported with a set of functions as per the Caterpillar performance handbook (1997). *ECA* has been implemented in MS Excel and automated using VBA. A numerical example was presented to demonstrate the use of *ECA* and its essential features.

EDA is a database that feeds the system with all equipment related information including their own characteristics and possible attachments. The database is essentially a relational database, designed using MS Access. Equipment data were extracted from the Caterpillar performance handbook (1997) and used to build the database. Earthmoving equipment were grouped into three major categories, according to the type of process they perform. These categories are load, haul, and support equipment. One piece of equipment may perform more than one process in different earthmoving operations. As a result, equipment can be shared among the three major categories of the application. *EDA* has been implemented utilizing MS Access 97 and its specially designed user interfaces have been coded utilizing VBA.

HTTA is a fuzzy clustering model for estimating haulers' travel time. Regression analysis has been performed to the significant factors that influence the travel time of haulers. Subsequently, three fuzzy clustering models have been developed (un-optimized, optimized Takagi-Sugeno 0^{th} order and optimized Takagi-Sugeno 1^{st} order) and tested. The optimized Takagi-Sugeno 1^{st} order out performed the others and was accordingly used for estimating haulers' travel time. *HTTA* provides a generic tool that can be incorporated in models dedicated for earthmoving operations. *HTTA* has been implemented utilizing VBA macros in MS Excel 97 environment.

EM_GA is a genetic algorithm, developed to search for a near-optimum fleet configuration that reduces project total cost. The developed algorithm accounts for the consideration of practical fleet configurations that account for equipment availability and incorporation of judgment and experience of contractors. *EM_GA* considers a set of qualitative and quantitative variables that influence the production of earthmoving operations. Qualitative variables represent the models of equipment used in each fleet scenario, whereas, quantitative variables represent the number of equipment involved in each scenario. It also accounts for the indirect cost portion associated with those operations. *EM_GA* has a powerful computational utility that speeds up the calculations by employing elitism and by storing the fitnesses of chromosomes in a built-up database to avoid duplication of fitness calculation. *EM_GA* has been implemented utilizing MS Visual C++6.0 and its dialog boxes have been coded in MS Visual Basic.

EMMA is a decision support environment for estimating markup. It is a generic application and is not limited to earthmoving. *EMMA* combines the advantages of multi-attribute utility theory and analytic hierarchy process. It also can be used to evaluate alternatives in different domains (e.g. engineering, procurement, and construction). The decision hierarchy is constructed in a flexible manner, according to the business environment. *EMMA* provides flexibility in capturing the decision-makers' attitudes towards risk. It has been coded utilizing MS visual basic 6.0 and is integrated with MS Excel to allow data import and export.

ORM has been developed to track the data processed by the rest of the system's components. It provides the user with necessary information for bid preparation. *ORM* consists of two main sub-modules: dynamic and static sub-modules. The dynamic sub-module extracts data only from the *EMSP* outputs, whereas, the static sub-module extracts data from all of the system's components. The dynamic sub-module supports animation to represent the overall system dynamics, the interaction between the customer and server resources, the bottlenecks of the system, and the delay associated with the resources involved. This sub-module utilizes Proof animation software (2000). On the other hand, the static sub-module provides output reports in a paper format pertaining to the selected fleet, equipment owning and operating cost details, simulation results, genetic algorithm results, project total duration, and project total cost.

8.2 Research Contributions

The contribution of this research can be summarized as:

1) The development of an object-oriented simulation model for estimating time and cost of earthmoving operations. The model accounts for the uncertainty involved in these operations and captures interaction amongst earthmoving equipment. It assists in selecting project fleets so as to minimize project total cost or duration, taking into account the availability of equipment.

2) The development of an optimization algorithm that provides near-optimum

fleet configurations that minimize project total cost.

3) The development of an automated environment for storing and processing data pertaining to the characteristics of earthmoving equipment including their owning and operating costs.

4) The development of a portable and efficient model for estimating haulers' travel time.

5) The developement of a decision support system, which can be used by contractors for estimating projects' markup and by owners for evaluating bid proposals, in a flexible manner.

6) The development of an integrated prototype software that facilitates bid preparation for earthmoving operations. It serves as a proof of concept for the developed methodology.

8.3 Recommendations for Future Research

This research has presented a methodology for optimizing earthmoving operations using computer simulation and genetic algorithms. The research can be expanded to account for the following:

1) Incorporating the methodology with actual data and knowledge gathered from the field data. This can be helpful in improving the methods used for

estimating equipment efficiencies and travel time of haulers. The field data can also be used to select the most representative probability distribution for the activities involved.

2) The optimization procedure can be expanded to include more than one criterion in the same time. For example, the objective function, in the optimization procedure, accounts for project duration and equipment idle time in addition to project total cost. This can be carried out, in a flexible manner, by providing a weight for each criterion in the objective function.

3) The simulation model and optimization algorithm can be extended to encompass different construction operations including tunneling, water and sewage pipelines, high-rise buildings, and highways.

REFERENCES

AbouRizk, S. M., and Hajjar, D. (1998). "A Framework for Applying Simulation in Construction." **Canadian Journal of Civil Engineering**, Vol. 25, No. 3, pp. 604-617.

AbouRizk, S. M., and Dozzi, S.P. (1993). "Application of Computer Simulation in Resolving Construction Disputes." **Journal of Construction Engineering and Management**, Vol. 119, No. 2, pp. 355-373.

AbouRizk, S. M., and Shi, J. (1994). "Automated Construction-Simulation Optimization." **Journal of Construction Engineering and Management**, Vol. 120, No. 2, pp. 374-385.

Abraham, D.M., and Halpin, D. W. (1998). "Simulation of the Construction of Cable-Stayed Bridges." **Canadian Journal of Civil Engineering**, Vol. 25, No. 3, pp. 490-499.

Ahmed, I., and Minkarah, I. A. (1988-a). "An Expert System for Selecting Bid Markups." **Proceedings of the Fifth Conference in Computing on Civil Engineering**, Virginia, U.S., pp. 229-238.

Ahmed, I., and Minkarah, I. A. (1988-b). "Questionnaire Survey on Bidding in Construction." **Journal of Management in Engineering**, Vol. 4, No. 3, pp. 229-243.

Ahmed, I., and Minkarah, I. A., (1987). "Optimum Markup for Bidding: a Preference-Uncertainty Trade off Approach." **Civil Engineering Systems**, Vol. 4, pp. 170-174.

Alkass, S., and Harris, F., (1988). "Expert System for Earthmoving Equipment Selection in Road Construction." **Journal of Construction Engineering and Management**, Vol. 114, No. 3, pp. 426-440.

Azadivar, F., and Tompkins, G. (1999). "Simulation Optimization with Qualitative Variables and Structural Model Changes: A Genetic Algorithm Approach." **European Journal of Operational Research**, Vol. 113, No. 1, pp. 169-182.

Azadivar, F. (1999). "Simulation Optimization Methodologies." **Proceedings of the 1999 Winter Simulation Conference**, Phoenix, AZ, U.S., pp. 198-204.

Azadivar, F., Shu, J., and Ahmed, M. (1996). "Simulation Optimization in Strategic Location of Semi-finished Product in a Pull-Type Production System." **Proceedings of the 1996 Winter Simulation Conference**, Coronado, CA, U.S., pp. 1123-1128.

Azadivar, F. (1992). "A Tutorial on Simulation Optimization." **Proceedings of the 1992 Winter Simulation Conference**, Arlington, VA, U.S., pp. 198-204.

Baesler, F. F., and Sepulveda, J. A. (2000). "Multi-Response Simulation Optimization Using Stochastic Genetic Search within a Goal Programming Framework." **Proceedings of the 2000 Winter Simulation Conference**, Orlando, FL, U.S., pp. 788-794.

Banks, J. (1998). **Handbook of Simulation**. John Wily & Sons, Inc., New York, NY.

Banks, J., Carson, J. S., Nelson, B. L., and Nicol, D. M. (2000). **Discrete-Event System Simulation**. Prentice Hall, Inc., New Jersey, NJ.

Bezdek, J. C, and Sankar, K. P. (1992). **Fuzzy Models for Pattern Recognition**. IEEE Publication, New York, NY.

Bezdek, J. C, Hathaway, R. J., Sabin, M. J., and Tucker, W. T. (1987). "Convergence Theory for Fuzzy c-Means: Counterexamples and Repairs." **IEEE Transactions on Systems, Man, and Cybernetics** Vol. SMC-17, No. 5, pp. 873-877.

Boesel, J., Nelson, B. L., and Ishii, N. (1999). **A Framework for Simulation-Optimization Software**. Technical Report, Department of Industrial Engineering and Management Science, Northwestern University.

Carr, R. I. (1982). "General Bidding Models." **Journal of Construction Division**, Vol. 108, No. CO4, pp. 639-650.

Carson, Y., and Maria, A. (1997). "Simulation Optimization: Methods and Applications." **Proceedings of the 1997 Winter Simulation Conference**, Atlanta, GA, U.S., pp. 118-126.

Carmichael, D. G. (1986-a). "Erlang Loading Models in Earthmoving." **Civil Engineering Systems**, Vol. 3, No. 3, pp. 118-124.

Carmichael, D. G. (1986-b). "Shovel-truck queues: a Reconciliation of Theory and Preactice." **Construction Management and Economics**, Vol. 4, No. 2, pp. 161-177.

Caterpillar Performance Handbook (1997). Edition 28, Caterpiller Inc., Peoria, IL.

Chan, W. T., Fwa, T. F., and Tan, C. Y. (1994). "Road Maintenance Planning using Genetic Algorithm: Formulation (Part1)." **Journal of Transportation Engineering**, Vol. 120, No. 5, pp. 693-709.

Chan, W. T., Chua, D. K. H., and Kannan, G. (1996). "Construction Resource Scheduling with Genetic Algorithms." **Journal of Construction Engineering and Management**, Vol. 122, No. 2, pp. 125-132.

Chao, L., and Skibniewski, M. J., (1994). "Estimating Construction Productivity: Neural-Network-Based Approach." **Journal of Construction Engineering and Management**, Vol. 8, No. 2, pp. 234-251.

Cheung, S. O., Lam, T. I., Leung, M. Y., and Wan, Y. W. (2001). "An Analytical Hierarchy Process Based Procurement Selection Method." **Construction Management and Economics**, Vol. 19, No. 4, pp. 427-437.

Chiu, S. L. (1994). "Fuzzy Model Identification Based on Cluster Estimation." **Journal of Intelligent and Fuzzy Systems**, Vol. 2, pp. 267-278.

Christian, J., and Xie, T. X. (1996). "Improving Earthmoving Estimation by More Realistic Knowledge." **Canadian Journal of Civil Engineering**, Vol. 23, No. 1, pp. 250-259.

Chua, D. K. H., and Li, D. (2000). "Key Factors in Bid Reasoning Model." **Journal of Construction Engineering and Management**, Vol. 126, No. 5, pp. 349-357.

Crain, R. C. (1997). "Simulation using GPSS/H." **Proceedings of the 1997 Winter Simulation Conference**, Atlanta, GA, U.S., pp. 567-573.

Coley, D. A. (1999). **An Introduction to Genetic Algorithms for Scientists and Engineers**. World Scientific, NJ.

Cor, H., and Martinez, J. C. (1999). "A case Study in the Quantification of a Change in the Conditions of a Highway Construction Operation." **Proceedings of the 1999 Winter Simulation Conference**, Phoenix, AZ, U.S., pp. 1007-1009.

Deitel, H. M., and Deitel, P. J. (1998). **C++ How to Program**. Prentice Hall, New Jersey, NJ.

Dozzi, S. P., AbouRizk, S. M., and Schroeder, S. L. (1996). "Utility-theory Model for Bid Markup Decisions." **Journal of construction Engineering and Management**, Vol. 122, No. 2, pp. 119-124.

Easa, S. M. (1987). "Earthwork Allocation with Nonconstant Unit Costs." **Journal of Construction Engineering and Management**, Vol. 113, No. 1, pp. 34-50.

EBC Web site (2001). **EBC Achievements: Civil Engineering/Earthworks** <http://www.ebcinc.qc.ca/>

Elmasri, R., and Navathe, S. B. (1994). **Fundamentals of Database Systems**. The Benjamin/Cummings Publishing Company, Inc., Redwood city, California.

EMF Users' Manual (1998). Caterpiller Inc., Peoria, IL.

Farid, F., and Koning, T. L. (1994). "Simulation Verifies Queuing Program for Selecting Loader-Truck Fleets." **Journal of Construction Engineering and Management**, Vol. 120, No. 2, pp. 386-404.

Feng, C., and Liu, L. (1997). "Using Genetic Algorithms to Solve Construction Time-Cost Trade-Off Problems." **Journal of Computing in Civil Engineering**, Vol. 11, No. 3, pp. 184-189.

Flood, I., and Christophilos, P. (1996). "Modeling Construction Processes Using Artificial Neural Networks." **Automation in Construction**, Vol. 4, No. 4, pp. 307-320.

Friedman, L. (1956). "A Competitive Bidding Strategy." **Operations Research**, Vol. 4, pp. 104-112.

FPC Users' Manual (1998). Caterpillar Inc., Peoria, IL.

Fuller, R. (2000). **Neuro-fuzzy Reasoning**. <http://www.abo.fi/~rfuller /robert.html>.

Fwa, T. F., Chan, W. T., and Tan, C. Y. (1995). "Optimal Programming by Genetic Algorithms for Pavement Management." **Transportation Research Record**, No. 1455, pp. 31-41.

Gates, M. (1967). "Bidding Strategies and Probabilities." **Journal of Construction Division**, Vol. 93, No. CO1, pp. 75-103.

Goldberg, D. E. (1989). **Genetic Algorithms in Search, Optimization and Machine Learning**. Addison-Wesley, Reading, Mass.

Gowda, R. K., Singh, A., and Connolly, M. (1998). "Holistic Enhancement of the Production Analysis of Bituminous Paving Operations." **Construction Management and Economics**, Vol. 16, No. 4, pp. 417-432.

Habib, S. (1996). "Computer Assisted Heavy Construction Bid Preparation." **Third Canadian Conference on Computing in Civil and Building Engineering**, pp. 227-239.

Haidar, A., Naoum, S., Howes, R., and Tah, J. (1999). "Genetic Algorithms Application and Testing for Equipment Selection." **Journal of Construction Engineering and Management**, Vol. 125, No. 1, pp. 32-38.

Hall, J. D., Bowden, R. O., and Usher, J. M. (1996). "Using Evaluation Strategies and Simulation to Optimize a Pull Production System." **Journal of Materials Processing Technology**, Vol. 61, pp. 47-52.

Halpin, D. W., and Riggs, L. S. (1992). **Planning and Analysis of Construction Operations**. John Wiley & Sons, Inc., New York, NY.

Halpin, D. W. (1977). "CYCLONE-Method for Modeling Job Site Processes." **Journal of Construction Division**, Vol. 103, No. 3, pp. 489-499.

Halpin, D. W., and Woodhead, R. W. (1976). **Design of Construction and Process Operations**. John Wiley & Sons, Inc., New York, NY.

Hajjar, D., and AbouRizk, S. M. (1999). "Simphony: An Environment for Building Special Purpose Simulation." **Proceedings of the 1999 Winter Simulation Conference**, Phoenix, AZ, U.S., pp. 998-1006.

Hason, S. F. (1994). **Feasibility and Implementation of Automation and Robotics in Canadian Building Construction Operation**. M.Sc.Thesis, Center for Building Studies, Concordia University.

Hatush, Z., and Skitmore, M. (1998). "Contractor Selection using Multicriteria Utility Theory: An Additive Model." **Building and Environment**, Vol. 33, No. 2, pp. 105-115.

Hegazy, T. (1999-a). "Optimization of Construction Time-Cost Trade-Off using Genetic Algorithms." **Canadian Journal of Civil Engineering**, Vol. 26, No. 6, pp. 685-697.

Hegazy, T. (1999-b). "Optimization of Resource Allocation and Leveling Using Genetic Algorithms." **Journal of Construction Engineering and Management**, Vol. 125, No. 3, pp. 167-175.

Hicks, J. C. (1993). "Haul-Unit Performance." **Journal of Construction Engineering and Management**, Vol. 119, No. 3, pp. 646-653.

Hicks, J. C. (1992). "Heavy Construction Estimates, with and without Computers." **Journal of Construction Engineering and Management**, Vol. 118, No. 3, pp. 545-560.

Holland, J. H. (1992). "Genetic Algorithms." **Scientific American**, July 1992, pp. 66-72.

Huang, R., Grigoriadis, A. M., and Halpin, D. W. (1994). "Simulation of Cable-Stayed Bridges using DISCO." **Proceedings of the 1994 Winter Simulation Conference**, Orlando, FL, U.S., pp. 1130-1136.

Hydro-Quebec Web site (2001). **Discover the New Sainte-Marguerite-3 Hydroelectric Project**. <http://www.hydroquebec.com/sm3_project/>.

Hydro-Quebec (1999). **Sainte-Marguerite-3 hydroelectric project: in harmony with the environment**.

Ioannou, P. G. (1999). "Construction of Dam Embankment with Nonstationary Queue." **Proceedings of the 1999 Winter Simulation Conference**, Phoenix, AZ, U.S., pp. 921-928.

Ioannou, P. G. (1989). **UM-CYCLONE user's manual**, Division of Construction Engineering and Management, Purdue University, West Lafayette, IN.

Jayawardane, A. K. W., and Harris, F. C. (1990) "Further Development of Integer Programming in Earthwork Optimization." **Journal of Construction Engineering and Management**, Vol. 116, No. 1, pp. 18-34.

Johnson, J. L. (1997). **Database: Models, Languages, Design**. Oxford University Press, New York, NY.

Joines, J. A., and Roberts, S. D. (1998). "Fundamentals of Object-Orientated Simulation." **Proceedings of the 1998 Winter Simulation Conference**, Washington, Washington DC, U.S., pp. 141-149.

Keeney, R. L., and Raiffa, H. (1993). **Decisions with Multiple Objectives: Preference and the Value Tradeoffs**. Cambridge University Press, New York, NY.

Keller, G., and Warrack, B. (1998). **Statistics for Management and Economics**. Brooks/Cole Publishing Company, Pacific Grove, CA.

Li, H., and Love, P. (1997). "Using Improved Genetic Algorithms to Facilitate Time-Cost Optimization." **Journal of Construction Engineering and Management**, Vol. 123, No. 3, pp. 233-237.

Lluch, J., and Halpin, D. W. (1982). "Construction Operations and Microcomputers." **Journal of Construction Division**, Vol. 108, CO1, pp. 129-145.

McCabe, B. (1998). "Belief Networks in Construction Simulation." **Proceedings of the 1998 Winter Simulation Conference**, Washington, Washington DC, U.S., pp. 1279-1286.

McCabe, B.Y. (1997). **An Automated Modeling Approach for Construction Performance Improvement Using Computer Simulation and Belief Networks**. Ph.D. Thesis, Alberta University.

Maria, D. (1997). "Introduction to Modeling and Simulation." **Proceedings of the 1997 Winter Simulation Conference**, Atlanta, GA, U.S., pp. 7-13.

Martinez, J. C., and Ioannou, P. J. (1999). "General-Purpose Systems for Effective Construction Simulation." **Journal of Construction Engineering and Management**, Vol. 125, No. 4, pp. 265-276.

Martinez, J. C. (1998). "EarthMover-Simulation tool for earthwork planning." **Proceedings of the 1998 Winter Simulation Conference**, Washington, Washington DC, U.S., pp. 1263-1271.

Martinez, J. C. (1996). **STROBOSCOPE-State and Resource Based Simulation of Construction Process**. Ph.D. Thesis, University of Michigan.

Marzouk, M., and Moselhi, O. (2001). "On the Use of Fuzzy Clustering in Construction Simulation." **Proceedings of the 2001 Winter Simulation Conference**, Arlington, VA, U.S., pp. 1547-1555.

Marzouk, M., and Moselhi, O. (2000). "Optimizing Earthmoving Operations using Object-Oriented Simulation." **Proceedings of the 2000 Winter Simulation Conference**, Orlando, FL, U.S., pp. 1926-1932.

Mayer, R. H., and Stark, M. S. (1981). "Earthmoving Logistics." **Journal of Construction Division**, Vol. 107, CO2, pp. 297-312.

Moselhi, O., and Marzouk, M. (2000). "Automated System for Cost Estimating of Earthmoving Operations." **17th International Symposium on Automation and Robotics in Construction**, Taipei, Taiwan, pp. 1053-1058.

Moselhi, O., Hegazy, T., and Fazio, P. (1993-a). "DBID: Analogy-Based DSS for Bidding in Construction." **Journal of Construction Engineering and Management**, Vol. 119, No. 3, pp. 466-476.

Moselhi, O., and Hegazy, T. (1993-b). "Markup Estimation using Neural Network Methodology." **Computing Systems in Engineering**, Vol. 4, No. 2-3, pp. 135-145.

Neter, J., and Wasserman, W. (1974). **Applied Linear Statistical Models**. Richard D. Irwin, Inc., Homewood, IL.

Nunnally, S. W. (1998). **Construction Methods and Management**. Prentice-Hall, Inc. Upper Saddle River, NJ.

Oloufa, A., Ikeda, M., and Nguyen, T. (1998). "Resource-Based Simulation Libraries for Construction." **Automation in Construction**, Vol. 7, No. 4, pp. 315-326.

Oloufa, A. (1993). "Modeling Operational Activities in Object-Oriented Simulation." **Journal of Computing in Civil Engineering**, Vol. 7, No. 1, pp. 94-106.

Paulson, G. C. Jr. (1978). "Interactive Graphics for Simulating Construction Operations." **Journal of Construction Division**, Vol. 104, No. 1, pp. 69-76.

Peer, G. A. (2001). "Ready to Serve." **Heavy Construction News**, March 2001, pp. 16-19.

Peer, G. A. (1998). "Powerhouse Takes Shape Far from Dam SM-3 Project." **Heavy Construction News**, March 2001, pp. 14-16.

Peurifoy, R. L., Ledbetter, W. B., and Schexnayder, C. J. (1996). **Construction Planning, Equipment, and Methods**. McGraw-Hill.Companies, Inc., New York, NY.

Philip, M., Mahadevan, N., and Varghese, K. (1997). "Optimization of Construction Site Layout A Genetic Algorithm Approach." **Proceedings of Fourth Congress of Computing in Civil Engineering**, pp. 710-717.

Pidd, M. (1995). "Object Orientation, Discrete Simulation and Three-Phase Approach." **Journal of Operation Research Society**, Vol. 46, No. 3, pp. 362-374.

Pidd, M. (1984). "Computer Simulation for Operational Research in 1984." **Developments in Operational Research**, R. W. Eglese and G. K. Rand, eds., Pregamon Press, Oxford, England, pp. 19-30.

Pritsker, A. A. B., O'Reilly, J. J., and LaVal, D. K. (1997). **Simulation with Visual SLAM and Awesim**. John Wiley & Sons, Inc., New York, NY.

Proof Animation Users' Manual (2000). Wolverine Software, Annandale, VA.

Quatrani, T. (1998). **Visual Modeling with Rational Rose and UML**, Addison-Wesley, Reading, Mass.

Roberts, C. A., and Dessouky, Y. M. (1998). "An overview of Object-Oriented Simulation." **Simulation**, Vol. 70, No. 6, pp. 359-368.

Saaty, T. L. (1982). **Decision Making for Leaders**. Wadsworth, Inc., California, CA.

Sawhney, A., and AbouRizk, S. M. (1996). "Computerized Tool for Hierarchical Simulation Modeling." **Journal of Computing in Civil Engineering**, Vol. 10, No. 2, pp. 115-124.

Sawhney, A., and AbouRizk, S. M. (1995). "HSM_Simulation-Based Planning Method for Construction Projects." **Journal of Construction Engineering and Management**, Vol. 121, No. 3, pp. 297-303.

Schriber, T. J., and Brunner, D. T. (1999). "Inside Discrete-Event Simulation Software: How it Works and how it Matters." **Proceedings of the 1999 Winter Simulation Conference**, Phoenix, AZ, U.S., pp. 72-80.

Seydel, J., and Olson, D. L. (1990). "Bids considering Multiple Criteria." **Journal of Construction Engineering and Management**, Vol. 116, No. 4, pp. 609-623.

Shannon, R. E. (1992). "Introduction to Simulation." **Proceedings of the 1992 Winter Simulation Conference**, Arlington, VA, U.S., pp. 65-73.

Shi, J., and AbouRizk, S. M. (1998). "An Automated Modeling system for Simulating Earthmoving Operations." **Computer-Aided Civil and Infrastructure Engineering**, Vol. 13, No. 1, pp. 121-130.

Shi, J., and AbouRizk, S. M. (1997). "Resource-Based Modeling for Construction Simulation." **Journal of Construction Engineering and Management**, Vol. 123, No. 1, pp. 26-33.

Shi, J., and AbouRizk, S. M. (1995). "An Optimization Method for Simulating Large Complex Systems." **Engineering Optimization**, Vol. 25, No. 3, pp. 213-229.

Shi, J. (1995). **Optimization for construction Simulation**. Ph.D. Thesis, Alberta University.

Skansholm, J. (1997). **C++ From the Beginning**, Addison-Wesley, Harlow, England.

Smith, S. D., Wood, G. S., and Gould, M. (2000). "A New Earthworks Estimating Methodology." **Construction Management and Economics**, Vol. 18, No. 2, pp. 219-228.

Smith, S. D. (1999). "Earthmoving Productivity Estimation Using Linear Regression Techniques." **Journal of Construction Engineering and Management**, Vol. 125, No. 3, pp. 133-141.

Smith, S. D., Osborne, J. R., and Forde, M. C. (1995). "Analysis of Earth-Moving systems Using Discrete-Event Simulation." **Journal of Construction Engineering and Management**, Vol. 121, No. 4, pp. 388-396.

Simphony Manual (2000). **Guide to the Development of Special Purpose Simulation Templates with Simphony**. Internal Report, Construction Engineering and Management, Department of Civil Engineering, University of Alberta.

Takagi, T., and Sugeno, M. (1985). "Fuzzy Identification of Systems and its Application to Modeling and Control." **IEEE Transactions on Systems, Man, and Cybernetics**, Vol. 15, pp. 116-132.

Tommelein, I. D. (1999). "Travel-Time Simulation to Locate and Staff Temporary Facilities under Changing Construction Demand." **Proceedings of the 1999 Winter Simulation Conference**, Phoenix, AZ, U.S., pp. 978-984.

Tommelein, I. D., Carr, R. I., and Odeh, A. M. (1994). "Assembly of simulation Networks Using Designs, Plans, and Methods." **Journal of Construction Engineering and Management**, Vol. 120, No. 4, pp. 796-815.

Touran, A. (1990). "Integration of Simulation with Expert Systems." **Journal of Construction Engineering and Management**, Vol. 116, No. 3, pp. 480-493.

Vanegas, J. A., Bravo, E. B., and Halpin, D. W. (1993). "Simulation Technologies for Planning Heavy Construction Processes." **Journal of Construction Engineering and Management**, Vol. 119, No. 2, pp. 336-354.

Visio Corporation (1997). Developing Visio Solutions, Scattle, Wa.

Wanous, M., Boussabaine, A. H., and Lewis, J. (2000). "To Bid or not to Bid: a Parametric Solution." **Construction Management and Economics**, Vol. 18, No. 4, pp. 457-466.

Yager, R. R., and Filev, D. P. (1994). "Approximate Clustering Via the Mountain Method." **IEEE Transactions on Systems, Man, and Cybernetics**, Vol. 24, No. 8, pp. 1279-1284.

APPENDIX A

CODES OF THE DEVELOPED SYSTEM COMPONENTS

```
//=============================================================================
//                          OPY_Simulate.h (Main Class)
//=============================================================================
#ifndef OPY_SIMULATE_H
#define OPY_SIMULATE_H
#include "EarthInfo.h"

class OPY_Simulate
{
public:
      OPY_Simulate(){;}
      OPY_Simulate(bool, bool, bool, double, int, int,bool);
      ~OPY_Simulate();
      void Set_Source();
      void Define_Activities(int,int,int);
      void Initiate_Simulation();
      void Load_Drive();
      void Haul_Drive();
      void Dump_Drive();
      void Return_Drive();
      void Activity_Drive();
      void Output_TotalDuration_Test();
      void Output_TotalDuration_Analysis();
      void Output_Earth_Quentity();
      void Output_TotalQuentity_Analysis();
      void Store_Info (double AA[], double);
      void Store_QueueLength_Statistics(double **, double **);
      void Store_WaitTime_Statistics(double **, double **);
      void Store_Activities_Duration_Statistics(double **, double **,
           double **, double **);
      void Store_SecondHauler_Activities_Statistics(double **, double **,
           double **, double **);
protected:
      enum ACTIVITIES {PILE=1, LOAD=2, HAUL=3, DUMP=4, RETURN=5, SPREAD=6,
           COMPACT=7};
      double mat;
      Hauled_Earth  h_earth;    //Moved Earth by Primary Activities
      int CrScen;
      int NH;                          //Number of hauler Models

      double HA_PL[2];
      int HA_NO[2];
      int Tot_Ha_No;                   //Total Number of haulers
      int NL;                          //Number of Loaders
      M_Queue QueueMag;

//    -------------------------------
//      Hauler Queue Info
//    -------------------------------
      Queue<Haul_Equip> HaulEquipmentQueue; //Queue of Haulers
      int HEQ_Length;                       //Hauler Queue Length
      double HEQ_WaitTime;                        //Hauler Queue Wait Time
      Haul_Equip *HEquip;                         //Matrix of Haulers
      Haul_Equip CrtHaulEquip; //Current Hauler
      double HE_QueueLength[5];//Queue to track the length of Haulers Queue
      double HE_QueueWaitTime[5];     //Queue to track the wait time in
           Haulers Queue

//    -------------------------------
//      Loader Queue Info
```

```
//   ------------------------------
      Queue<Load_Equip> LoadEquipmentQueue; //Queue of Loaders
      int LEQ_Length;                       //Loader Queue Length
      double LEQ_WaitTime;                        //Loader Queue Wait Time
      Load_Equip *LEquip;                         //Matrix of Loaders
      Load_Equip CrtLoadEquip;              //Current Loader
      double LE_QueueLength[5];//Queue to track the length of Loaders Queue
      double LE_QueueWaitTime[5];     //Queue to track the wait time in
           Loaders Queue

//   ------------------------------
//    Load Activity Dur Info
//   ------------------------------
      double Load1_Act_Dur[5];        //Queue to track Load1 Activity
           Duration
      double Load2_Act_Dur[5];        //Queue to track Load2 Activity
           Duration

//   ------------------------------
//    Haul Activity Dur Info
//   ------------------------------
      double Haul1_Act_Dur[5];        //Queue to track Load1 Activity
           Duration
      double Haul2_Act_Dur[5];        //Queue to track Load1 Activity
           Duration

//   ------------------------------
//    Dump Activity Dur Info
//   ------------------------------
      double Dump1_Act_Dur[5];        //Queue to track Load1 Activity
           Duration
      double Dump2_Act_Dur[5];        //Queue to track Load1 Activity
           Duration

//   ------------------------------
//    Return Activity Dur Info
//   ------------------------------
      double Return1_Act_Dur[5];            //Queue to track Load1 Activity
           Duration
      double Return2_Act_Dur[5];            //Queue to track Load1 Activity
           Duration
      double curtime;                 //Current Simulation Time
      double ActDur;                        //Activity Duration
      Activity  Act;
      M_Activity  Actmag;
      Activity_List ActList;
      int Activity_Index;      //TO obtain the first Activity Indicator in
           the list
      double CrtPayLoad; //Current Hauler PayLoad
      int Released_Loader_ID;
      int Released_Hauler_ID;
      int Released_Hauler_Model;
      double Total_Primary_Activities_Dur;
      bool TERM_Time, TERM_Quent, TestMode, Animat_Mode;
      double Termination_Time;
      int CurtRun;
      int NoRun;
};
#endif
```

```
//=============================================================================
//                          OPE_Simulate.h (Main Class)
//=============================================================================
#ifndef OPE_SIMULATE_H
#define OPE_SIMULATE_H
#include "OPY_SIMULATE.h"
#include "EarthInfo.h"

class OPE_Simulate : public virtual OPY_Simulate
{
public:
      OPE_Simulate(){;}
      OPE_Simulate(bool, bool, bool, double, int, int,bool);
      ~OPE_Simulate();
      void Set_Source();
      void Define_Activities(int,int,int);
      void Initiate_Simulation();
      void Pile_Drive();
      void Load_Drive();
      void Haul_Drive();
      void Dump_Drive();
      void Return_Drive();
      void Activity_Drive();
      void Output_Earth_Quentity();
      void Output_TotalDuration_Test();
      void Output_TotalDuration_Analysis();
      void Output_TotalQuentity_Analysis();
      void Store_QueueLength_Statistics(double **, double **, double **);
      void Store_WaitTime_Statistics(double **, double **, double **);
      void Store_Activities_Duration_Statistics(double **, double **,
          double **, double **,double **);
protected:
      Piled_Earth  p_earth;           //Moved Earth by Pile Equipment
      int NP;                         //Total Number Pile Equip
      double P_Prod;                  //Productivity Pile Equip

//      Pile Queue Info
//    -------------------------------
      Queue<Pile_Equip> PileEquipmentQueue; //Queue of Pile Equips
      int PEQ_Length;                       //Pile Equips Queue Length
      double PEQ_WaitTime;                        //Pile Equips Queue Wait
Time
      Pile_Equip *PEquip;                         //Matrix of Pile Equips
      Pile_Equip CrtPileEquip;              //Current Pile Equip
      double PE_QueueLength[5];//To track the length of Pile Equips Queue
      double PE_QueueWaitTime[5];//To track the wait time in Pile Equips
Queue

//     Pile Activity Dur Info
//    -------------------------------
      double Pile_Act_Dur[5];   //Queue to track Pile Activity Duration
      int Released_Pile_Equip_ID;
      double Total_Pile_Activity_Dur;
};
#endif
```

```
//=============================================================================
//                        OSD_Simulate.h (Main Class)
//=============================================================================
#ifndef OSD_SIMULATE_H
#define OSD_SIMULATE_H
#include "OPY_SIMULATE.h"
#include "EarthInfo.h"

class OSD_Simulate : public virtual OPY_Simulate
{
public:
      OSD_Simulate(){;}
      OSD_Simulate(bool, bool, bool, double, int, int,bool);
      ~OSD_Simulate();
      void Set_Source();
      void Define_Activities(int,int,int);
      void Initiate_Simulation();
      void Load_Drive();
      void Haul_Drive();
      void Dump_Drive();
      void Return_Drive();
      void Spread_Drive();
      void Activity_Drive();
      void Output_TotalDuration_Test();
      void Output_TotalDuration_Analysis();
      void Output_Earth_Quentity();
      void Output_TotalQuentity_Analysis();
      void Store_QueueLength_Statistics(double **, double **, double **);
      void Store_WaitTime_Statistics(double **, double **, double **);
      void Store_Activities_Duration_Statistics(double **, double **,
           double **, double **,double **);
protected:
      Spreaded_Earth      s_earth;    //Spreaded Earth by Spread Equipment
      int NS;                         //Total Number Spread Equip
      double S_Prod;                  //Productivity Spread Equip

//     Spread Queue Info
//   -------------------------------
      Queue<Spread_Equip> SpreadEquipmentQueue;    //Queue of Spread Equips
      int SEQ_Length;                          //Spread Equips Queue Length
      double SEQ_WaitTime;                          //Spread Equips Queue
           Wait Time
      Spread_Equip *SEquip;                         //Matrix of Spread Equips
      Spread_Equip CrtSpreadEquip;                  //Current Spread Equip
      double SE_QueueLength[5];//To track the length of Spread Equips Queue
      double SE_QueueWaitTime[5];     //To track the wait time in Spread
           Equips Queue

//    Spread Activity Dur Info
//   -------------------------------
      double Spread_Act_Dur[5];//Queue to track Spread Activity Duration
      int Released_Spread_Equip_ID;
      double Total_Spread_Activity_Dur;
};
#endif
```

```
//=============================================================================
//                          OCT_Simulate.h (Main Class)
//=============================================================================
#ifndef OCT_SIMULATE_H
#define OCT_SIMULATE_H
#include "OPY_SIMULATE.h"
#include "EarthInfo.h"

class OCT_Simulate : public virtual OPY_Simulate
{
public:
      OCT_Simulate(){;}
      OCT_Simulate(bool, bool, bool, double, int, int,bool);
      ~OCT_Simulate();
      void Set_Source();
      void Define_Activities(int,int,int);
      void Initiate_Simulation();
      void Load_Drive();
      void Haul_Drive();
      void Dump_Drive();
      void Return_Drive();
      void Compact_Drive();
      void Activity_Drive();
      void Output_TotalDuration_Test();
      void Output_TotalDuration_Analysis();
      void Output_Earth_Quentity();
      void Output_TotalQuentity_Analysis();
      void Store_QueueLength_Statistics(double **, double **, double **);
      void Store_WaitTime_Statistics(double **, double **, double **);
      void Store_Activities_Duration_Statistics(double **, double **,
           double **, double **,double **);
protected:
      Compacted_Earth c_earth;        //Compacted Earth by Spread Equipment
      int NC;                               //Total Number Compact Equip
      double C_Prod;                        //Productivity Compact Equip

//      Compact Queue Info
//    ------------------------------
      Queue<Compact_Equip> CompactEquipmentQueue;//Queue of Compact Equips
      int CEQ_Length;                       //Compact Equips Queue Length
      double CEQ_WaitTime;                        //Compact Equips Queue
           Wait Time
      Compact_Equip *CEquip;                //Matrix of Compact Equips
      Compact_Equip CrtCompactEquip;        //Current Compact Equip
      double CE_QueueLength[5];//To track the length of Compact Equips
           Queue
      double CE_QueueWaitTime[5];     //To track the wait time in Compact
           Equips Queue

//     Compact Activity Dur Info
//    ------------------------------
      double Compact_Act_Dur[5];      //Queue to track Compact Activity
           Duration
      int Released_Compact_Equip_ID;
      double Total_Compact_Activity_Dur;
};
#endif
```

```
//=============================================================================
//                          PS_Simulate.h (Main Class)
//=============================================================================
#ifndef PS_SIMULATE_H
#define PS_SIMULATE_H
#include "OPE_SIMULATE.h"
#include "OSD_SIMULATE.h"
#include "EarthInfo.h"

class PS_Simulate  : public OPE_Simulate, public OSD_Simulate
{
public:
      PS_Simulate(bool, bool, bool, double, int, int,bool);
      ~PS_Simulate();
      void Set_Source();
      void Define_Activities(int,int,int);
      void Initiate_Simulation();
      void Pile_Drive();
      void Load_Drive();
      void Haul_Drive();
      void Dump_Drive();
      void Return_Drive();
      void Spread_Drive();
      void Activity_Drive();
      void Output_TotalDuration_Test();
      void Output_TotalDuration_Analysis();
      void Output_Earth_Quentity();
      void Output_TotalQuentity_Analysis();
      void Store_QueueLength_Statistics(double **, double **, double **,
           double **);
      void Store_WaitTime_Statistics(double **, double **, double **,
           double **);
      void Store_Activities_Duration_Statistics(double **, double **,
           double **, double **,double **, double **);
};
#endif
```

```
//=============================================================================
//                          PC_Simulate.h (Main Class)
//=============================================================================
#ifndef PC_SIMULATE_H
#define PC_SIMULATE_H
#include "OPE_SIMULATE.h"
#include "OCT_SIMULATE.h"
#include "EarthInfo.h"

class PC_Simulate  : public OPE_Simulate, public OCT_Simulate
{
public:
      PC_Simulate(bool, bool, bool, double, int, int,bool);
      ~PC_Simulate();
      void Set_Source();
      void Define_Activities(int,int,int);
      void Initiate_Simulation();
      void Pile_Drive();
      void Load_Drive();
      void Haul_Drive();
      void Dump_Drive();
      void Return_Drive();
      void Compact_Drive();
      void Activity_Drive();
      void Output_TotalDuration_Test();
      void Output_TotalDuration_Analysis();
      void Output_Earth_Quentity();
      void Output_TotalQuentity_Analysis();
      void Store_QueueLength_Statistics(double **, double **, double **,
           double **);
      void Store_WaitTime_Statistics(double **, double **, double **,
           double **);
      void Store_Activities_Duration_Statistics(double **, double **,
           double **, double **,double **, double **);
};
#endif
```

```
//=============================================================================
//                          SC_Simulate.h (Main Class)
//=============================================================================
#ifndef SC_SIMULATE_H
#define SC_SIMULATE_H
#include "OSD_SIMULATE.h"
#include "OCT_SIMULATE.h"
#include "EarthInfo.h"

class SC_Simulate : public OSD_Simulate, public OCT_Simulate
{
public:
      SC_Simulate(){;}
      SC_Simulate(bool, bool, bool, double, int, int,bool);
      ~SC_Simulate();
      void Set_Source();
      void Define_Activities(int,int,int);
      void Initiate_Simulation();
      void Load_Drive();
      void Haul_Drive();
      void Dump_Drive();
      void Return_Drive();
      void Spread_Drive();
      void Compact_Drive();
      void Activity_Drive();
      void Output_TotalDuration_Test();
      void Output_TotalDuration_Analysis();
      void Output_Earth_Quentity();
      void Output_TotalQuentity_Analysis();
      void Store_QueueLength_Statistics(double **, double **, double **,
           double **);
      void Store_WaitTime_Statistics(double **, double **, double **,
           double **);
      void Store_Activities_Duration_Statistics(double **, double **,
           double **, double **,double **, double **);
};
#endif
```

```
//=============================================================================
//                          PSC_Simulate.h (Main Class)
//=============================================================================
#ifndef PSC_SIMULATE_H
#define PSC_SIMULATE_H
#include "OPE_SIMULATE.h"
#include "OSD_SIMULATE.h"
#include "OCT_SIMULATE.h"
#include "EarthInfo.h"

class PSC_Simulate : public OPE_Simulate, public OSD_Simulate , public
            OCT_Simulate
{
public:
       PSC_Simulate(){;}
       PSC_Simulate(bool, bool, bool, double, int, int,bool);
       ~PSC_Simulate();
       void Set_Source();
       void Define_Activities(int,int,int);
       void Initiate_Simulation();
       void Pile_Drive();
       void Load_Drive();
       void Haul_Drive();
       void Dump_Drive();
       void Return_Drive();
       void Spread_Drive();
       void Compact_Drive();
       void Activity_Drive();
       void Output_TotalDuration_Test();
       void Output_TotalDuration_Analysis();
       void Output_Earth_Quentity();
       void Output_TotalQuentity_Analysis();
       void Store_QueueLength_Statistics(double **, double **, double **,
            double **, double **);
       void Store_WaitTime_Statistics(double **, double **, double **,
            double **, double **);
       void Store_Activities_Duration_Statistics(double **, double **,
            double **, double **,double **, double **, double **);
};
#endif
```

```
//=============================================================================
//                        SPT_Simulate.h (Auxiliary Class)
//=============================================================================
#ifndef SPT_SIMULATE_H
#define SPT_SIMULATE_H
#include "OPY_SIMULATE.h"
#include "EarthInfo.h"

class SPT_Simulate
{
public:
      SPT_Simulate(int, bool, bool, bool, double,int, int,bool);
      ~SPT_Simulate();
      void Set_Source();
      void Define_Activities(int);
      void Store_Info (double AA[], double);
      void Initiate_Simulation();
      void Support_Drive();
      void Activity_Drive();
      void Output_Earth_Quentity();
      void Output_TotalDuration_Test();
      void Output_TotalQuentity_Analysis();
      void Output_TotalDuration_Analysis();
      void Store_QueueLength_Statistics(double **);
      void Store_WaitTime_Statistics(double **);
      void Store_Activities_Duration_Statistics(double **);

protected:
      double mat;
      int CurrentActivity;
      Processed_Earth  p_earth; //Moved Earth by Pile Equipment
      int NST;                  //Total Number Support Equip
      double ST_Prod;           //Productivity Support Equip
      int CrScen;
      M_Queue QueueMag;
      double curtime;    //Current Simulation Time
      double ActDur;            //Activity Duration
      Activity  Act;
      M_Activity  Actmag;
      Activity_List ActList;
      double Total_Sec_Act_Dur;
      bool TERM_Time, TERM_Quent, TestMode, Animat_Mode;
      double Termination_Time;

//    -------------------------------
//     Support Queue Info
//    -------------------------------
      Queue<Support_Equip> SupportEquipmentQueue; //Queue of Support Equips
      int STQ_Length;                          //Support Equips Queue Length
      double STQ_WaitTime;                           //Support Equips Queue
           Wait Time
      Support_Equip *STEquip;                  //Matrix of Support Equips
      Support_Equip CrtSupportEquip;           //Current Support Equip


//    -------------------------------
//     Support Activity Dur Info
//    -------------------------------
      int Released_Support_Equip_ID;
      double Support_Act_Dur[5];      //To track Load1 Activity Duration
```

```
        double ST_QueueWaitTime[5];     //To track the wait time in Support
             Equips Queue
        double ST_QueueLength[5];//To track the length of Support Equips
             Queue
        int CurtRun;
        int NoRun;
};
#endif
```

```
//=============================================================================
//                          M_Simulate.h (Auxiliary Class)
//=============================================================================
#ifndef M_SIMULATE_H
#define M_SIMULATE_H
#include "OPY_Simulate.h"
#include "OPE_Simulate.h"

class M_Simulate
{
public:
      M_Simulate(){;}
      M_Simulate(bool, bool, int, int, int, int, double, bool);
      ~M_Simulate();
      void Output_Statistics(int, double**, fstream &fout);
      void Simulate_Operations();
      void Output_Statistics(fstream &fout);
      void Output_Simulation_Results();

private:
      double Total_NSamples, Total_SampleSum, Total_SampleSqSum, Maximum,
           Minimum ;
      double MEAN, SD;

      int InvolvedActivities;
      int CurrentCaseScenario;
      int NoOfRuns;
      int SecondHauler;
      int Working_Minutes;
      bool Animation_Mode;

      double **PE_LENGTH_ARRAY;
      double **HE_LENGTH_ARRAY;
      double **LE_LENGTH_ARRAY;
      double **SE_LENGTH_ARRAY;
      double **CE_LENGTH_ARRAY;

      double **PE_W_TIME_ARRAY;
      double **HE_W_TIME_ARRAY;
      double **LE_W_TIME_ARRAY;
      double **SE_W_TIME_ARRAY;
      double **CE_W_TIME_ARRAY;

      double **PILE_DUR_ARRAY;
      double **LOAD1_DUR_ARRAY, **LOAD2_DUR_ARRAY;
      double **HAUL1_DUR_ARRAY, **HAUL2_DUR_ARRAY;
      double **DUMP1_DUR_ARRAY, **DUMP2_DUR_ARRAY;
      double **RETURN1_DUR_ARRAY, **RETURN2_DUR_ARRAY;
      double **SPREAD_DUR_ARRAY;
      double **COMPACT_DUR_ARRAY;

      bool M_Test_Mode, M_TERM_Time, M_TERM_Quent, M_Equip_Interaction;
      double M_Termination_Time;
};
#endif
```

```
//=============================================================================
//                          Activity.h (Auxiliary Class)
//=============================================================================
#ifndef ACTIVITY_LIST_H
#define ACTIVITY_LIST_H

class Activity
{
public:
      void Set_Activity_Attributes(double ST, double DUR, int A_ID, int
           E_ID, int E_MD, int L_ID)
      {
           StartTime=ST;
           Duration=DUR;
           FinishTime=StartTime+Duration;
           ActivityID=A_ID;
           EquipID=E_ID;
           EquipModel=E_MD;
           LoaderID=L_ID;
      }

      double StartTime;
      double FinishTime;
      double Duration;
      int ActivityID;//1>Pile, 2>Load, 3>Haul, 4>Dump, 5>Return, 6>Spread,
           7>Compact
      int EquipID; //According to the no of used equipment (1,2,     .... )
      int EquipModel;//According to the no of equipment models:  1 or 2
           For Haulers
      int LoaderID;      //Used in Case of Load Activity (I.e ActivityID=2)
                  // Loader No       For Load Activity
                  // 0               For Others
};
```

```
//=============================================================================
//                          Activity_List.h (Auxiliary Class)
//=============================================================================
class AcElement
{
      friend class Activity_List;           // make List a friend
public:
      //constructor
      AcElement (){;}
      AcElement (const Activity&Dsimactivity)
      {
            simactivity=Dsimactivity;
            nextPtr=0 ;
      }
      Activity get_Activity () const {return simactivity;}

private:
      Activity simactivity;
      AcElement *nextPtr;                   // next node in the list
};

class Activity_List
{
public:
      Activity_List();   // constructor
      ~Activity_List();  // denstructor
      void Add_Activity_To_List (const Activity&);
      void Remove_Activity_From_List();
      Activity Get_First_Activity() ;
      bool AcisEmpty() const ;
      void Acprint() const ;

      //Animation
      void Set_Trace_Objects(bool,int,int,double);

private:
      void Add_Top_Activity (const Activity&);
      void Add_End_Activity (const Activity&);
      int NoEl;
      AcElement *firstPtr;      // pointer to first node
      AcElement *lastPtr;       // pointer to last node
      int CONT;
      Activity SAC[100]; // Temporary Activity List
      double *PTAL;
      AcElement *ptr, *newPtr, *tempPtr, *currentPtr;
      AcElement *TempPtr1,*TempPtr2;
      AcElement * getNewNode(const Activity Ssimactivity); //To allocate a
           new node
      //Animation
      bool Is_Ani_Exist;
      Trace_Animation  TAnimate;
      void Trace_Add_Manager(int,int,int,double,double);
      void Trace_Remove_Manager(int,int,int,double,double);
};
#endif
```

```
//=============================================================================
//                          Queue.h (Auxiliary Class)
//=============================================================================
#include <iostream.h>
#include <iomanip.h>
#include <fstream.h>
#include <assert.h>

#ifndef QUEUE_H
#define QUEUE_H

template<class TYPE> class Queue;
template<class TYPE>
class Element {
      friend class Queue <TYPE>;            // make List a friend
public:
      //constructor
      Element (const TYPE &info)
      {
           data=info;
           nextPtr=0 ;
      }
      TYPE getData () const {return data;}

private:
      TYPE data;
      Element<TYPE> *nextPtr;         // next node in the list

};


template <class TYPE>
class Queue
{
public:
      Queue();                  // constructor
      ~Queue();                 // destructor
      void EnQueue(const TYPE&);
      bool DeQueue();
      bool isEmpty() const;
      int Get_Number_Of_Elements();
      TYPE Get_info() ;

private:
      Element<TYPE> *firstPtr; // pointer to first node
      Element<TYPE> *lastPtr;  // pointer to last node

      // Utility function to allocate a new node
      Element<TYPE> *getNewNode(const TYPE&);
      int No_Element;           //The no of the hauler in the list

      Element<TYPE> *currentPtr, *tempPtr, *newPtr, *ptr;
};
#endif
```

```
//=============================================================================
//                          M_Queue.h (Auxiliary Class)
//=============================================================================
#ifndef M_QUEUE_H
#define M_QUEUE_H
#include "Equipment.h"
#include "Queue.h"

class M_Queue
{
public:

      Queue();

      template <class T>
      void QueueIn(Queue <T> &ElementObjects, T value)
      {
           ElementObjects.EnQueue(value);
      }

      template <class T>
      int Get_Length(Queue <T> &ElementObjects)
      {
           return ElementObjects.Get_Number_Of_Elements();
      }

      template <class T>
      T Get_Frist_Element(Queue <T> &ElementObjects )
      {
           return ElementObjects.Get_info();
      }

      template <class T>
      void QueueOut(Queue <T> &ElementObjects)
      {
           ElementObjects.DeQueue();
      }

      template <class T>
      bool IsQueueEmpty(Queue <T> &ElementObjects)
      {
           return ElementObjects.isEmpty();
      }
};
#endif
```

```
//=============================================================================
//                          Equipment.h (Auxiliary Class)
//=============================================================================
#ifndef EQUIPMENT_H
#define EQUIPMENT_H
#include <iostream.h>
#include <iomanip.h>
#include <fstream.h>
#include <assert.h>

class Equipment
{
public:
      void Set_Equip_Attributes(int EN, double ET)
      {
           Equip_No= EN;
           Entrance_Time = ET;
      }
      int Equip_No;
      double Entrance_Time;
};


class Haul_Equip: public Equipment
{
public:
      void Set_Equip_Attributes(int HT, int EN, double PL, double ET)
      {
           Hauler_Type=HT;
           Equip_No= EN;
           PayLoad=PL;
           Entrance_Time = ET;
      }
      int Hauler_Type;
      double PayLoad;
};
#endif
```

```
//=============================================================================
//                          Piled_Earth.h (Auxiliary Class)
//=============================================================================
class Piled_Earth
{
public:
      //constructor
      Piled_Earth(){Piled_Soil=0, Earth_To_Pile=0, Ready_To_Haul=0;}
      void Set_Earth_To_Pile (double
           SEarth_To_Pile){Earth_To_Pile=SEarth_To_Pile;}
      void Increase_PiledEarth(double ExcavatorCapacity)
      {
           Ready_To_Haul +=ExcavatorCapacity;
           Piled_Soil +=ExcavatorCapacity;
      }
      void Decrease_PiledEarth(double HaulerCapacity)
                                {Ready_To_Haul -=HaulerCapacity;}
      bool IsHaulPossible (double HaulerCapacity)
      {
      if (Ready_To_Haul < HaulerCapacity)
            return false;
           else
            return true;
      }
      bool IsPileCompleted ()
           {
           if (Piled_Soil < Earth_To_Pile)
            return false;
           else
            return true;
           }
      double Return_Prepared_Earth(){return Piled_Soil;}

private:
      double Earth_To_Pile;
      double Ready_To_Haul;
      double Piled_Soil;
};
```

```
//=============================================================================
//                          Hauled_Earth.h (Auxiliary Class)
//=============================================================================
class Hauled_Earth
{
public:
      //constructor
      Hauled_Earth(){Hauled_Soil=0, Dumped_Soil= 0, Earth_To_Haul=0,
           Ready_To_SP_CP=0;}
      void Set_Earth_To_Haul(double
           SEarth_To_Haul){Earth_To_Haul=SEarth_To_Haul;}
      void Increase_HauledEarth(double HaulerCapacity)
            {Hauled_Soil +=HaulerCapacity;}
      void Increase_DumpedEarth(double HaulerCapacity)
      {
           Ready_To_SP_CP +=HaulerCapacity;
           Dumped_Soil +=HaulerCapacity;
      }
      void Decrease_HauledEarth(double SP_CP_Cap){Ready_To_SP_CP -
           =SP_CP_Cap;}
      bool IsSp_CoPossible (double Sp_CoCap)
      {
           if (Ready_To_SP_CP < Sp_CoCap)
             return false;
           else
             return true;
      }
      bool IsHaulCompleted ()
      {
           if (Hauled_Soil < Earth_To_Haul)
             return false;
           else
             return true;
      }
      bool IsDumpCompleted ()
      {
           if (Dumped_Soil < Earth_To_Haul)
             return false;
           else
             return true;
      }

      double Return_Prepared_Earth(){return Dumped_Soil;}
      double Return_H_Earth(){return Hauled_Soil;}
      double Return_S_Earth(){return Ready_To_SP_CP;}

private:
      double Hauled_Soil;
      double Dumped_Soil;
      double Earth_To_Haul;
      double Ready_To_SP_CP;
};
```

```
//=============================================================================
//                          Spread_Earth.h (Auxiliary Class)
//=============================================================================
class Spread_Earth
{
public:
      //constructor
      Spreaded_Earth(){Spreaded_Soil=0, Earth_To_Spread=0, Ready_To_CP=0;}
      void Set_Earth_To_Spread(double
           SEarth_To_Spread){Earth_To_Spread=SEarth_To_Spread;}
      void Increase_SpreadedEarth(double SpreaderCap)
      {
           Ready_To_CP +=SpreaderCap;
           Spreaded_Soil +=SpreaderCap;
      }
      void Decrease_SpreadedEarth(double CP_Cap){Ready_To_CP -=CP_Cap;}
      bool Is_CoPossible (double CoCap)
      {
           if (Ready_To_CP < CoCap)
             return false;
           else
             return true;
      }
      bool IsSpreadCompleted ()
      {
           if (Spreaded_Soil < Earth_To_Spread)
             return false;
           else
             return true;
      }
      double Return_Prepared_Earth(){return Spreaded_Soil;}
      double Return_P_Earth(){return Ready_To_CP;}


private:
      double Spreaded_Soil;
      double Earth_To_Spread;
      double Ready_To_CP;
};
```

```
//=============================================================================
//                          Compacted_Earth.h (Auxiliary Class)
//=============================================================================
class Compacted_Earth
{
public:
      //constructor
      Compacted_Earth(){Compacted_Soil=0, Earth_To_Compact=0;}
      void Set_Earth_To_Compact(double
           SEarth_To_Compact){Earth_To_Compact=SEarth_To_Compact;}
      void Increase_CompactedEarth(double CompactorCap)
      {
           Compacted_Soil +=CompactorCap;
      }
      bool IsCompactCompleted ()
      {
      if (Compacted_Soil < Earth_To_Compact)
           return false;
      else
           return true;
      }
      double Return_Prepared_Earth(){return Compacted_Soil;}

private:
      double Compacted_Soil;
      double Earth_To_Compact;
};
```

```
//=============================================================================
//                          Trace_Animation.h (Auxiliary Class)
//=============================================================================
#include <iostream.h>
#include <iomanip.h>
#include <fstream.h>
#include <assert.h>
#ifndef TRACE_ANIMATION_H
#define TRACE_ANIMATION_H

class Trace_Animation{
public:
      Trace_Animation();
      void Set_Current_Time(double);
      void Set_Hauler_Capacity(double HC){HaulerCap=HC;}
      void Create(int,int);
      void Destroy(int);
      void Place_At(int,double,double);
      void Place_On(int,int);
      void Place_At_End(int,int);
      void Change_Message();
      void Set_Object_Travel(int,double);
      void End_Trace();

private:
      double Trace_Time;
      double HaulerCap;
      int DumpedEarth;
      fstream OUT_TRACE;
};
#endif
```

```
//=============================================================================
//          Mathematical Modeling Code for Activity Distribution
//=============================================================================
double M_Activity::GET_DISTRIBUTION(double DP[])
                                  //DP stands for Distribution Parameters
{
   switch (int (DP[0]))
   {

//     DETERMINSTIC
//---------------------
       case (1):
       {
            return DP[1];
            break;
       }

//     UNIFORM  DISTRIBUTION
//-----------------------------
       case (2):
       {
            return (DP[1]+(DP[2]-DP[1])*RND);
            break;
       }

//     TRIANGLE  DISTRIBUTION
//-----------------------------
       case (3):
       {
            R=RND;
            if (R<=(DP[2]-DP[1])/(DP[3]-DP[1]))
            return DP[1]+sqrt((DP[2]-DP[1])*(DP[3]-DP[1])*R);
            else
            return DP[3]-sqrt((DP[3]-DP[2])*(DP[3]-DP[1])*(1-R));
            break;
       }

//     NORMAL  DISTRIBUTION
//-----------------------------
       case (4):
       {
            do {
            RR1=2*(RND)-1;
            RR2=2*(RND)-1;
            W=RR1*RR1+RR2*RR2;
            }while(W>1);

            //It generats pairs, to select one of them
            //if a generated random number less than 0.5, pick the first,
            otherwise the second
            if ((RND)<0.5){
            return (RR1 *(sqrt(-2*log(W)/W))*DP[2]+DP[1]);}
            else{
            return (RR2*(sqrt(-2*log(W)/W))*DP[2]+DP[1]);}
            break;
       }


//     EXPONENTIAL  DISTRIBUTION
//--------------------------------
```

```
        case (5):
        {
             return (-DP[1]*log(RND));
             break;
        }


//      ERLANG  DISTRIBUTION
//--------------------------------
        case (6):
        {
             LogSum=0;
             for(int i=1; i<=int(DP[2]);i++)
             {
             RE=RND;
             LogSum=LogSum+log(RE);
             }
             return (-DP[1]*LogSum);
             break;
        }


//      LOGNORMAL  DISTRIBUTION
//--------------------------------
        case (7):
        {
             SEGMA2=log(((DP[2]*DP[2])/(DP[1]*DP[1]))+1);
             NEW=(log (DP[1])-0.5*(SEGMA2*SEGMA2));
               TEMP1[0]=4;
               TEMP1[1]=NEW;
               TEMP1[2]=SEGMA2;
               return GET_DISTRIBUTION(TEMP1);
             break;
        }

//      GAMMA  DISTRIBUTION
//--------------------------------
        case (8):
        {
             double BETA, ALFA;
             BETA=((DP[2]*DP[2])/DP[1]);
             ALFA=(DP[1]/BETA);

             // CASE (0<ALFA<1)
             if ((ALFA>0)&&(ALFA<1))
             {
               double X, Y;
               X=0;
               Y=0;
             do {
               X=pow(RND, (1/ALFA));
               Y=pow(RND, (1/(1-ALFA)));
               }while ((X+Y)>1);

               WG=X/(X+Y);
               return (-WG*log(RND)*BETA);
             }

             // CASE (1<=ALFA<5)
             else if ((ALFA>=1)&&(ALFA<5))
```

```
            {
              A=int(ALFA);
              B=ALFA-A;
            do {
              LogS=0;
              for(int i=0; i<A;i++)
              {
              LogS+=log(RND);
              }
              XG=(ALFA/A)*(-LogS);
              }while (RND>(pow((XG/ALFA),B)*exp(-B*(XG/ALFA-1))));
              return (XG*B);
            }

            // CASE (ALFA=>5)
            else
            {
              R=RND;
              INT_ALFA=int(ALFA);
              double TEMP2[3];
                     TEMP2[0]=6;
                     TEMP2[1]=BETA;
              if (R>=(ALFA-int(ALFA))){
                     TEMP2[2]=INT_ALFA;
                     return GET_DISTRIBUTION(TEMP2);}
              else{
                     TEMP2[2]=INT_ALFA+1;
                     return GET_DISTRIBUTION(TEMP2);}
            }
      break;
      }

//    BETA  DISTRIBUTION
//-------------------------
      case (9):
      {
            TMEAN=(DP[1]-DP[3])/(DP[4]-DP[3]);
            TVARIANCE=pow((DP[2]/(DP[4]-DP[3])),2);

            ALPHA=((TMEAN*TMEAN)/TVARIANCE)*(1-TMEAN)-TMEAN;
            B_BETA=ALPHA*((1-TMEAN)/TMEAN);

                     TEMP3[0]=8;
                     TEMP3[1]=1;
                     TEMP3[2]=ALPHA;
            G1=GET_DISTRIBUTION(TEMP3);
                     TEMP3[2]=B_BETA;
            G2=GET_DISTRIBUTION(TEMP3);
            return ((G1/(G1+G2))*(DP[4]-DP[3])+DP[3]);
      break;
      }

//    DETERMINSTIC
//---------------------
      default:
            return DP[1];
   }
}
```

```
//=============================================================================
//                              Genetic Algorithm
//=============================================================================
#ifndef GENETIC_ALGORTHIM
#define GENETIC_ALGORTHIM

class Genetic_Algorthims {
public:
      Genetic_Algorthims (int,int,int,int,double,bool, bool, bool, int,
           int, double, double, int,int);
      ~Genetic_Algorthims ();
      void Set_Ranges();
      void InitiatePopulation();
      void Mutate();
      void SelectEliteCromosomes();
      void PreformElitism();
      void CrossOver (int, int, int);
      void NoCrossOver (int, int, int);
      void Update_TempArrays();
      void Update_PopulationArray();
      void IntiateFitness();
      void SetFitness();
      int EstimateFitness(int);
      bool StoreGenerationsStatistics();
      void ExportGenerationsStatistics();
      void ShowDetails();
      int Selction();


private:
      //Memeber Functions
      int Round_Number (double);
      void FitnessStatistics();
      int Combine_Genes(int);
      void Store_DataBase_Info(int, int);
      int Check_Existance_Database(int);

      //Attributes
      int TotalNoOfScenario, CurrentOperation;
      int NoOfGenes;
      int **GeneMaxNo, **GeneMinNo, **UserDefined;
      double **EquipHCost;
      double **EquipCapacities;
      int *TempChromosome;
      int **ChromosomeFitnessDataBase;
      int RetrivedDBFitness,SearchIndex;

      int **PopulationArray, **TempPopulationArray, **TempSenarioArray;
      int **ElitChromosomeArray;
      double *FitnessArray,*TempFitnessArray,*FitnessSum, *FitnessAverage,
           *BestFitness;
      double *ElitFitnessArray;
      double *RecentFitnessArray;
      double *ModifiedTempArray, ModifiedSum, TempNorm;
      int *BestFitnessRank;
      double **Output_Pop_Fitness,**Output_Fitness_Stat;
      int PopulationSize, TotalPopulationSize;
      int MaxNoOfGenerations;
      double MutationProb;
      bool SecHauler,ExtremeIndicator,UserDefConfg;
```

```
        int CrossPosition, MutPosition;
        int CrossConuter, MutConuter, DatabaseConuter,
             BestFitnessRankConuter;
        double Sum, RouletteWheel;
        int SelectedChromosome;
        int CurrentIndex;
        int CurrentGeneration, CurrentCombination;;
        int CombinedCromosome;
        int DailyHours, MonthlyDays;
        double Intial_IC_Sum, Monthly_IC;
        double MaterialQuentity;
        int NoOfRuns;
        int NoImprovLimit;
        int Gen_EstimatedFitness;
        double SimulationOutput[3];
        double TotalCost;
};
#endif
```

```
'=============================================================================
'                    Estimating the Potential of Data Points
'=============================================================================
Sub EstimatePotential()
Dim k As Single
Dim str, str1, str2, str3 As String
str1 = "=R"
str3 = "C[-6]-RC[-6]"
For i = 1 To 180
      Range("N4").Select
      str2 = CStr(i + 3)
      str = str1 + str2 + str3
      ActiveCell.FormulaR1C1 = str
      Range("N4").Select
      Selection.Copy
      Range("N5:N183").Select
      ActiveSheet.Paste
      Range("O4:Q183").Select
      ActiveSheet.Paste
      Range("S4").Select
      Application.CutCopyMode = False
      ActiveCell.FormulaR1C1 = _
           "=(RC[-5]*RC[-5])+(RC[-4]*RC[-4])+(RC[-3]*RC[-3])+(RC[-2]*RC[-
           2])"
      Range("S4").Select
      Selection.Copy
      Range("S5:S183").Select
      ActiveSheet.Paste
      Application.CutCopyMode = False
      Range("T4").Select
      ActiveCell.FormulaR1C1 = "=EXP(-9.467*RC[-1])"
      Selection.Copy
      Range("T5:T183").Select
      ActiveSheet.Paste
      Range("T4:T184").Select
      Range("T184").Activate
      ActiveCell.FormulaR1C1 = "=SUM(R[-180]C:R[-1]C)"
      Application.CutCopyMode = False
      k = Cells(184, 20)
      Cells(3 + i, 13) = k
Next i
```

```
'=========================================================================
'            Estimating the Travel Time Using OTS_1 (Haul Trip)
'=========================================================================
Function AverageSpeed(SegementPosition, P_MaxSpeed, C_Length, C_MaxSpeed)
'Switch Segment Type
'-------------------
'SegementPosition
'F      First Segement
'L      Last Segement
'M      Last Segement
'S      Single Segement

N_Length = (C_Length - 5) / (5000 - 5)
N_MaxSpeed = (C_MaxSpeed - 5.86) / (67.06 - 5.86)

Select Case SegementPosition
'**************************

Case ("M")
'---------
    N_P_MaxSpeed = (P_MaxSpeed - 5.86) / (67.06 - 5.86)
    Memship_1 = Exp(-9.467 * ((N_P_MaxSpeed - 0.1098) ^ 2 + (N_Length -
          0.4597) ^ 2 + (N_MaxSpeed - 0.1098) ^ 2))
    Memship_2 = Exp(-9.467 * ((N_P_MaxSpeed - 0.1709) ^ 2 + (N_Length -
          0.5225) ^ 2 + (N_MaxSpeed - 0.7987) ^ 2))

    TS_Function_1 = -0.001967 * P_MaxSpeed - 0.000145 * C_Length + 0.981433
          * C_MaxSpeed + 0.493252
    TS_Function_2 = 0.077631 * P_MaxSpeed + 0.002859 * C_Length + 0.795181
          * C_MaxSpeed - 2.514986

    AverageSpeed = ((Memship_1 * TS_Function_1) + (Memship_2 *
          TS_Function_2)) / (Memship_1 + Memship_2)

Case ("S")
'---------
    Memship_1 = Exp(-9.467 * ((N_Length - 0.5413) ^ 2 + (N_MaxSpeed -
          0.142) ^ 2))
    Memship_2 = Exp(-9.467 * ((N_Length - 0.4783) ^ 2 + (N_MaxSpeed -
          0.8176) ^ 2))
    Memship_3 = Exp(-9.467 * ((N_Length - 0.0166) ^ 2 + (N_MaxSpeed -
          0.1044) ^ 2))

    TS_Function_1 = 0.000281 * C_Length + 0.933059 * C_MaxSpeed - 0.730892
    TS_Function_2 = 0.005921 * C_Length + 0.728397 * C_MaxSpeed - 8.277534
    TS_Function_3 = 0.003528 * C_Length + 0.867688 * C_MaxSpeed - 0.866201

    AverageSpeed = ((Memship_1 * TS_Function_1) + (Memship_2 *
          TS_Function_2) + (Memship_3 * TS_Function_3)) / (Memship_1 +
          Memship_2 + Memship_3)

Case ("F")
'---------
    Memship_1 = Exp(-9.467 * ((N_Length - 0.5588) ^ 2 + (N_MaxSpeed -
          0.133) ^ 2))
    Memship_2 = Exp(-9.467 * ((N_Length - 0.2228) ^ 2 + (N_MaxSpeed -
          0.7569) ^ 2))

    TS_Function_1 = 0.000349 * C_Length + 0.854412 * C_MaxSpeed + 0.46132
    TS_Function_2 = 0.006139 * C_Length + 0.346664 * C_MaxSpeed + 13.807516
```

```
    AverageSpeed = ((Memship_1 * TS_Function_1) + (Memship_2 *
           TS_Function_2)) / (Memship_1 + Memship_2)

Case ("L")
'---------
    Memship_1 = Exp(-9.467 * ((N_Length - 0.4609) ^ 2 + (N_MaxSpeed -
           0.1381) ^ 2))
    Memship_2 = Exp(-9.467 * ((N_Length - 0.7572) ^ 2 + (N_MaxSpeed -
           0.7982) ^ 2))

    TS_Function_1 = 0.000111 * C_Length + 0.975071 * C_MaxSpeed - 0.09899
    TS_Function_2 = 0.004664 * C_Length + 0.858164 * C_MaxSpeed - 12.977785

    AverageSpeed = ((Memship_1 * TS_Function_1) + (Memship_2 *
           TS_Function_2)) / (Memship_1 + Memship_2)
End Select
'**********

End Function
```

```
'=============================================================================
'            Estimating the Travel Time Using OTS_1 (Return Trip)
'=============================================================================
Function AverageSpeed(SegementPosition, P_MaxSpeed, C_Length, C_MaxSpeed)
'Switch Segment Type
'-------------------
'SegementPosition
'F       First Segement
'L       Last Segement
'M       Last Segement
'S       Single Segement

N_Length = (C_Length - 5) / (5000 - 5)
N_MaxSpeed = (C_MaxSpeed - 16.32) / (67.06 - 16.32)

Select Case SegementPosition
'**************************

Case ("M")
'-------------------------
    N_P_MaxSpeed = (P_MaxSpeed - 16.32) / (67.06 - 16.32)

    Memship_1 = Exp(-9.467 * ((N_P_MaxSpeed - 0.1817) ^ 2 + (N_Length -
          0.3033) ^ 2 + (N_MaxSpeed - 0.1817) ^ 2))
    Memship_2 = Exp(-9.467 * ((N_P_MaxSpeed - 0.4176) ^ 2 + (N_Length -
          0.6068) ^ 2 + (N_MaxSpeed - 0.7793) ^ 2))
    Memship_3 = Exp(-9.467 * ((N_P_MaxSpeed - 0.7793) ^ 2 + (N_Length -
          0.4675) ^ 2 + (N_MaxSpeed - 0.1425) ^ 2))
    Memship_4 = Exp(-9.467 * ((N_P_MaxSpeed - 0.0836) ^ 2 + (N_Length -
          0.1079) ^ 2 + (N_MaxSpeed - 0.7572) ^ 2))

    TS_Function_1 = 0.026636 * P_MaxSpeed + 0.000086 * C_Length + 0.87981 *
          C_MaxSpeed + 1.572828
    TS_Function_2 = 0.098879 * P_MaxSpeed + 0.001133 * C_Length + 0.959156
          * C_MaxSpeed - 6.806374
    TS_Function_3 = 0.006076 * P_MaxSpeed - 0.00005 * C_Length + 0.998029 *
          C_MaxSpeed - 0.264956
    TS_Function_4 = 0.401315 * P_MaxSpeed + 0.010683 * C_Length + 0.535142
          * C_MaxSpeed + 0.368654

     AverageSpeed = ((Memship_1 * TS_Function_1) + (Memship_2 *
          TS_Function_2) + (Memship_3 * TS_Function_3) + (Memship_4 *
          TS_Function_4)) / (Memship_1 +   Memship_2 + Memship_3 +
          Memship_4)

Case ("S")
'---------
    Memship_1 = Exp(-9.467 * ((N_Length - 0.4945) ^ 2 + (N_MaxSpeed -
          0.1927) ^ 2))
    Memship_2 = Exp(-9.467 * ((N_Length - 0.5596) ^ 2 + (N_MaxSpeed -
          0.9267) ^ 2))
    Memship_3 = Exp(-9.467 * ((N_Length - 0.0597) ^ 2 + (N_MaxSpeed -
          0.7793) ^ 2))

    TS_Function_1 = 0.000568 * C_Length + 0.890406 * C_MaxSpeed + 0.367658
    TS_Function_2 = 0.003711 * C_Length + 1.032054 * C_MaxSpeed - 20.712072
    TS_Function_3 = 0.022888 * C_Length + 0.404144 * C_MaxSpeed + 1.996745
    AverageSpeed = ((Memship_1 * TS_Function_1) + (Memship_2 *
          TS_Function_2) + (Memship_3 * TS_Function_3)) / (Memship_1 +
          Memship_2 + Memship_3)
```

```
Case ("F")
'---------
    Memship_1 = Exp(-9.467 * ((N_Length - 0.4989) ^ 2 + (N_MaxSpeed -
           0.206) ^ 2))
    Memship_2 = Exp(-9.467 * ((N_Length - 0.5197) ^ 2 + (N_MaxSpeed -
           0.757) ^ 2))

    TS_Function_1 = -0.000116 * C_Length + 0.927247 * C_MaxSpeed + 1.597802
    TS_Function_2 = 0.003581 * C_Length + 0.764984 * C_MaxSpeed - 1.198001

    AverageSpeed = ((Memship_1 * TS_Function_1) + (Memship_2 *
           TS_Function_2)) / (Memship_1 + Memship_2)

Case ("L")
'---------
    Memship_1 = Exp(-9.467 * ((N_Length - 0.4981) ^ 2 + (N_MaxSpeed -
           0.186) ^ 2))
    Memship_2 = Exp(-9.467 * ((N_Length - 0.357) ^ 2 + (N_MaxSpeed -
           0.7583) ^ 2))

    TS_Function_1 = 0.000154 * C_Length + 0.802667 * C_MaxSpeed + 3.531259
    TS_Function_2 = 0.003904 * C_Length + 0.53547 * C_MaxSpeed + 12.918018

    AverageSpeed = ((Memship_1 * TS_Function_1) + (Memship_2 *
           TS_Function_2)) / (Memship_1 + Memship_2)
End Select
'**********

End Function
```

```
'=============================================================================
'                          Static Sub-Module Functions
'=============================================================================
Sub SetMaxMini()
NoOFRuns = Worksheets("INFO").Cells(3, 6)
For j = 0 To 14
Worksheets("INFO").Cells(1002, 12 + j) = ""
Worksheets("INFO").Cells(1003, 12 + j) = ""
Worksheets("INFO").Cells(1006, 12 + j) = ""
Worksheets("INFO").Cells(1007, 12 + j) = ""
If (Worksheets("INFO").Cells(2, 12 + j) = "") Then
    GoTo FollowingColumnEnd:
End If
MaximumValue = Worksheets("INFO").Cells(2, 12 + j)
MinimumValue = Worksheets("INFO").Cells(2, 12 + j)
    For i = 2 To NoOFRuns
    If (MinimumValue > Worksheets("INFO").Cells(i + 1, 12 + j)) Then
        MinimumValue = Worksheets("INFO").Cells(i + 1, 12 + j)
    End If
    If (MaximumValue < Worksheets("INFO").Cells(i + 1, 12 + j)) Then
        MaximumValue = Worksheets("INFO").Cells(i + 1, 12 + j)
    End If
    Next i
    Worksheets("INFO").Cells(1006, 12 + j) = CDbl(MinimumValue)
    Worksheets("INFO").Cells(1007, 12 + j) = MaximumValue
FollowingColumnEnd:
Next j
End Sub


Function SetDurationExpectedValue()
NoOFRuns = Worksheets("INFO").Cells(3, 6)
ReDim TempArray(NoOFRuns)
' Fill the Matrix
For j = 0 To 14
TempSumAvg = 0
TempAvg = 0
For i = 1 To NoOFRuns
    If (Worksheets("INFO").Cells(i + 1, 12 + j) = "") Then
        GoTo FollowingColumn:
    End If
   TempSumAvg = TempSumAvg + Worksheets("INFO").Cells(i + 1, 12 + j)
Next i
    Worksheets("INFO").Cells(1003, 12 + j) = TempSumAvg / NoOFRuns
    If (CInt(TempSumAvg / NoOFRuns) - TempSumAvg / NoOFRuns < 0) Then
        TempAvg = CInt(TempSumAvg / NoOFRuns) + 1
    Else
        TempAvg = CInt(TempSumAvg / NoOFRuns)
    End If
    Worksheets("INFO").Cells(1002, 12 + j) = TempAvg
FollowingColumn:
Next j
End Function
```

```
'================================================================================
'                          Dynamic Sub-Module Trace File
'================================================================================
Create  LOADER  1
Place  1 On  LoaderQueue at end
Create  TRUCK_R  11
Place  11 On  HaulerQueue at end
Create  TRUCK_R  12
Place  12 On  HaulerQueue at end
Create  TRUCK_R  13
Place  13 On  HaulerQueue at end
Destroy  11
Create  TRUCK_H  11
Place  1 On  LoaderWay
Place  11  440  204
Time  3.41264
Destroy  11
Place  1 On  LoaderQueue at end
Create  TRUCK_F  11
Place  11 On  HaulRoad
set  11  travel  24.9718
Destroy  12
Create  TRUCK_H  12
Place  1 On  LoaderWay
Place  12  440  204
Time  6.99752
Destroy  12
Place  1 On  LoaderQueue at end
Create  TRUCK_F  12
Place  12 On  HaulRoad
set  12  travel  24.7695
Destroy  13
Create  TRUCK_H  13
Place  1 On  LoaderWay
Place  13  440  204
Time  10.2936
Destroy  13
.
.
.
.
Time  432.783
Place  12 On  HaulerQueue
Time  436.609
Place  13 On  HaulerQueue
Time  457.083
Destroy  11
Create  TRUCK_D  11
Place  11  308  160
write Quantity  504
Time  458.636
Destroy  11
End
```

In order to allow users to review input data and edit it as they see fit, two batch files have been generated as external files. The first batch file (EMSP.bat) is utilized to run the simulation engine and it has the following fields:

Field No.	Data Type
1	Defined path
2	Either simulation is performed for a test mode or an analysis mode
3	Termination condition of for the simulation analysis
4	Interaction is allowed among equipment or not
5	Storage of simulation data for later use in animation
6	Selected fleet scenario for simulation analysis
7	Existence of second hauler model
8	Define involved activities in the simulation process
9	Number of simulation runs.

The second batch file (EM_GA.bat) is utilized to run the developed genetic algorithm and it has the following fields:

Field No.	Data Type
1	Defined path
2	Define involved activities in the simulation process
3	Sub-population size
4	Number of fleet scenarios
5	Number of generations
6	Number of simulation runs
7	Non-improvement limit
8	Crossover probability
9	Mutation probability
10	Existence of second hauler model
11	Is elitism applicable
12	Is extreme fleet configurations applicable
13	Is user-defined fleet configurations applicable
14	Scheduled daily hours
15	Number of working days per month
16	Time-independent indirect cost ($)
17	Time-related indirect cost ($/month)

APPENDIX B

SIMEARTH: GENERATED REPORTS

SIMULATION STATISTICS

Scenario 1

Project Name: Stage 1, Moraine Fill

Expected Total Project Time (Hrs.): 181

Queue Information

Queue Length

Queue Name	Mean	SD	Maximum	Minimum
Hauler	0.50	0.51	3.00	0
Loader	0.00	0.04	1.00	0
Excavator	NA	NA	NA	NA
Spreader	0.50	0.50	1.00	0
Compactor	0.50	0.50	1.00	0

Queue Wait Time (min.)

Queue Name	Mean	SD	Maximum	Minimum
Hauler	0.07	0.33	4.91	0.00
Loader	13.49	10.79	40.46	0
Excavator	NA	NA	NA	NA
Spreader	6.03	9.62	37.08	0
Compactor	4.24	8.33	36.51	0

Activities Durations (min.)

Primary

	Activity Name	Mean	SD	Maximum	Minimum
Hauler Type 1	Load	2.29	0.12	2.50	2.10
	Haul	21.75	0.68	23.57	20.38
	Dump	2.04	0.09	2.20	1.90
	Return	18.42	0.58	19.97	17.26
Hauler Type 2	Load	NA	NA	NA	NA
	Haul	NA	NA	NA	NA
	Dump	NA	NA	NA	NA
	Return	NA	NA	NA	NA

Secondary

Activity Name	Mean	SD	Maximum	Minimum
Excavate	NA	NA	NA	NA
Spread	2.12	0.09	2.28	1.97
Compact	1.53	0.07	1.65	1.42

DIRECT COST REPORT

Scenario 1

Project Name: Stage 1, Moraine Fill

Expected Project Duration (Hrs.): 181

Fleet Elements Cost Breakdown

Haulers

Model	No.	$ / Hr.	Operating Hours	Total Cost ($)
777D	3	212.95	181	115,632
NA	NA	NA	NA	0
Total Haulers Cost				115,632

Loaders

Model	No.	$ / Hr.	Operating Hours	Total Cost ($)
992G	1	295.74	181	53,529
Total Loaders Cost				53,529

Excavators

Model	No.	$ / Hr.	Operating Hours	Total Cost ($)
NA	NA	NA	NA	0
Total Excavators Cost				0

Spreaders

Model	No.	$ / Hr.	Operating Hours	Total Cost ($)
D8R	1	153.51	181	27,785
Total Spreaders Cost				27,785

Compactors

Model	No.	$ / Hr.	Operating Hours	Total Cost ($)
CS-583C	1	89.62	181	16,221
Total Compactors Cost				16,221

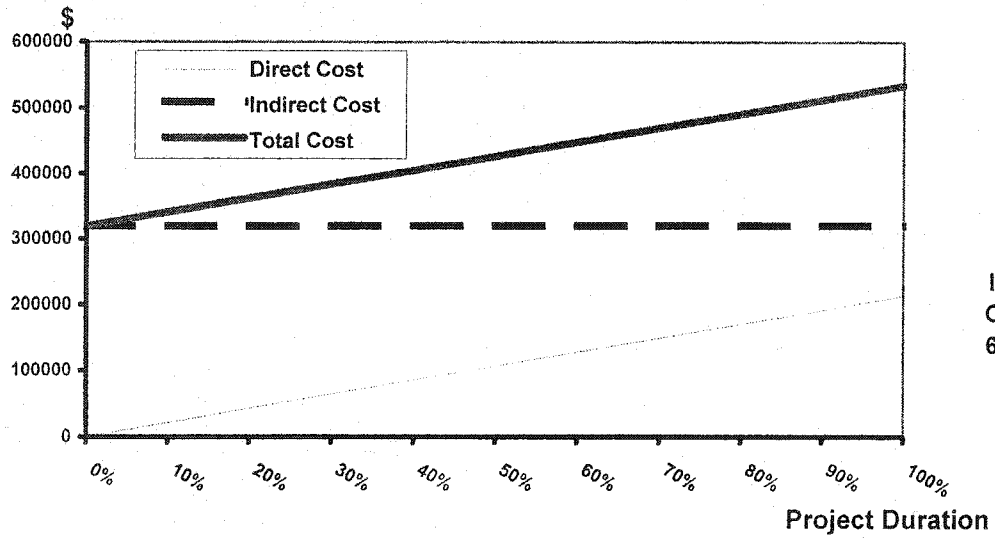
Cost Summary

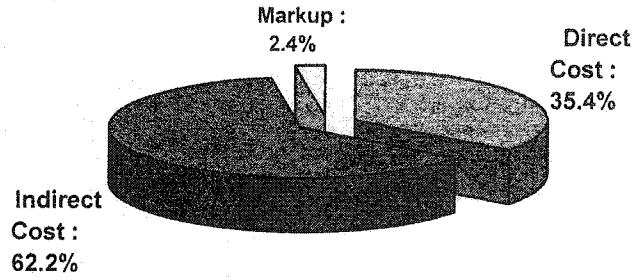
Equiment	Cost ($)
Haulers	115,632
Loaders	53,529
Excavators	0
Spreaders	27,785
Compactors	16,221
Total Direct Cost	213,167

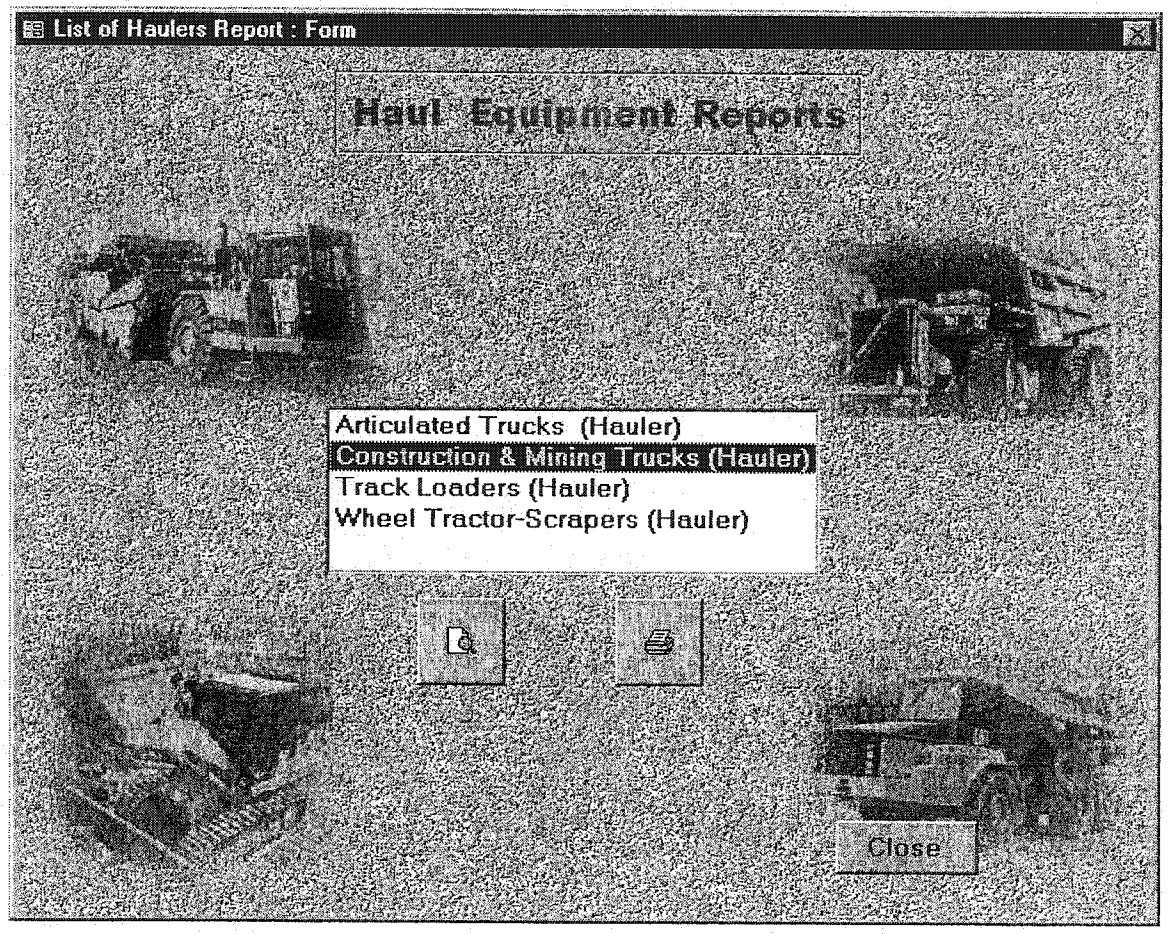
TOTAL COST Breakdown Structure

Scenario 1

Project Name :		Stage 1, Moraine Fill				
Direct Cost :		213,167	$	Total Cost :	587,867	$
Indirect Cost :		374,700	$	Bid Amount :	602,564	$
Markup :	2.5 %	14,697	$	Cost / Unit :	20.65	$/CM

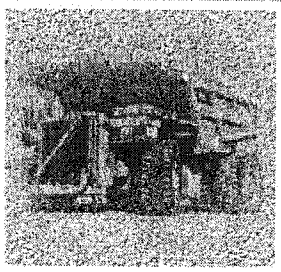





List of Haulers Report : Form
Haul Equipment Reports
Articulated Trucks (Hauler)
Construction & Mining Trucks (Hauler)
Track Loaders (Hauler)
Wheel Tractor-Scrapers (Hauler)
Close

Haul Equipment Report

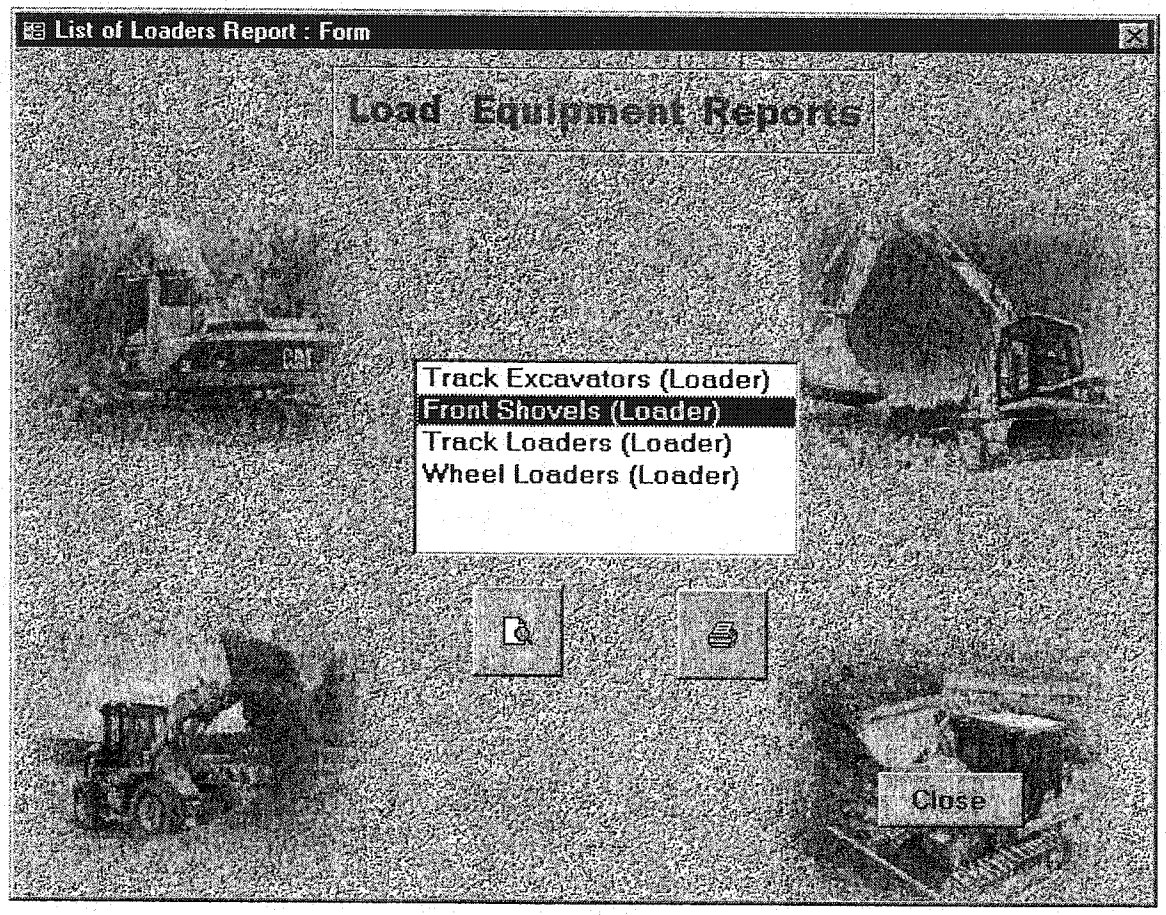
Construction and Mining Trucks



Truck Model : 773D

Truck Characteristics:

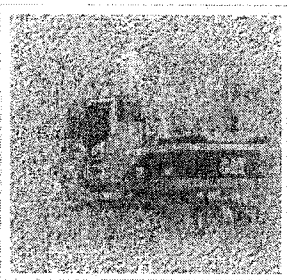
Flywheel (KW)	:	485
Engine Model	:	3412E
No. of Cylinders	:	12
Empty Weight (Ton)	:	40.2
Rated Load (Ton)	:	52.3
Top Speed (Km/hr)	:	66
Standard Tires	:	24.00R35(E-4)
Tank Capacity (L)	:	700
Truck Cost ($/hr)	:	$161.00


List of Loaders Report : Form
Load Equipment Reports
Track Excavators (Loader)
Front Shovels (Loader)
Track Loaders (Loader)
Wheel Loaders (Loader)
Close

Load Equipment Report

Front Shovels

Shovel Model / Source : 5130B (U.S.)



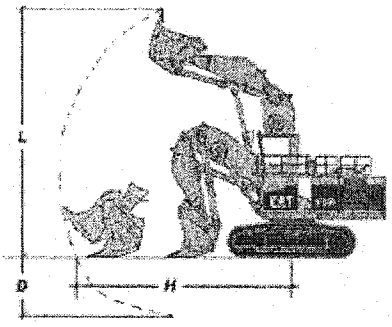
Shovel Characteristics:

Flywheel (KW)	:	597
OperatingWeight (Ton)	:	181
Engine Model	:	3508BEUI
Rated Engine (RPM)	:	1750
No. of Cylinders	:	8
Tank Capacity (L)	:	2600

Bucket Specifications:

Bucket Type	:	General Purpose
bucket Width(m)	:	3.64
Bucket Capacity (CM)	:	11
Bucket Weight(Ton)	:	15.79

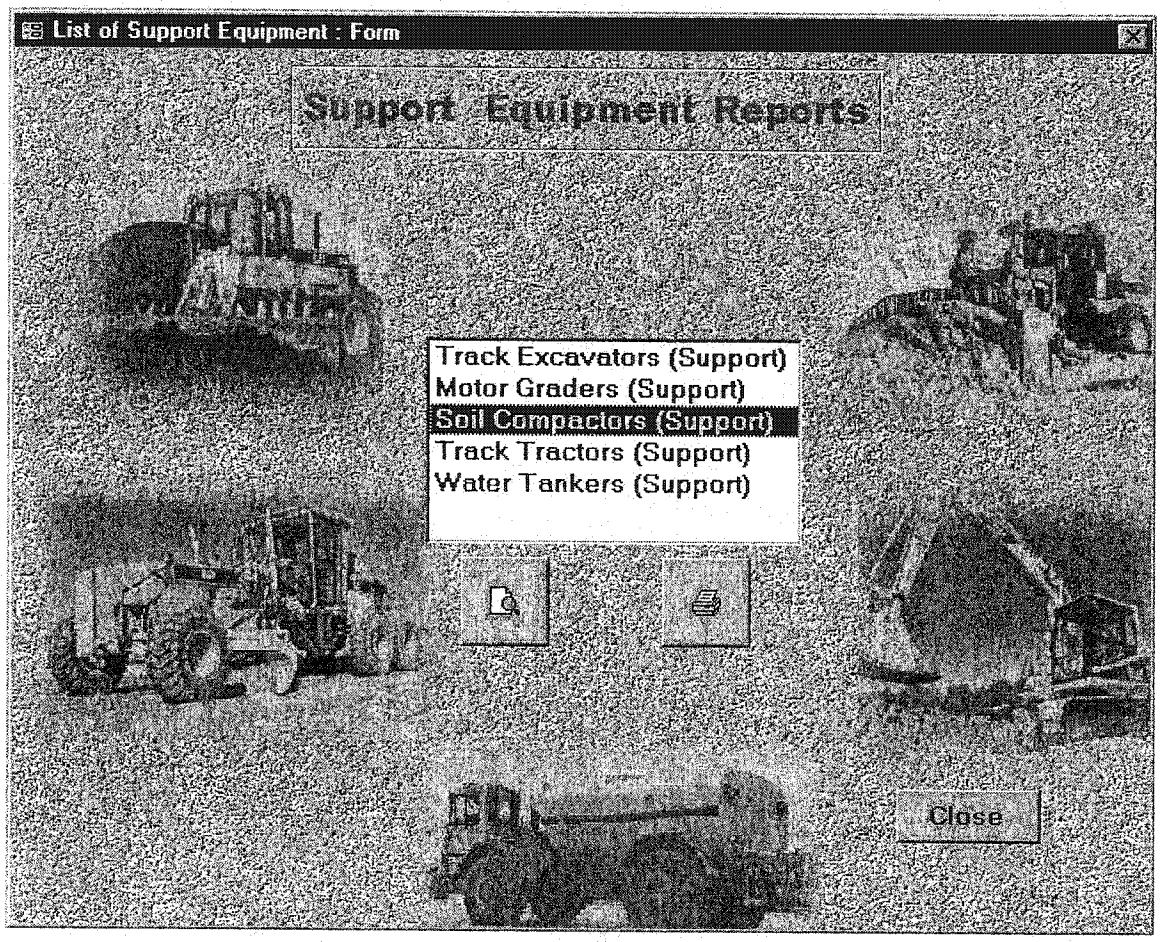
Range Dimentions:



Maximum Loading Height (L)	:	13.4
Groung Reach (H)	:	11.7
Maximum Digging Depth (D)	:	3.14

**Hint: All Dimensions are in meters*

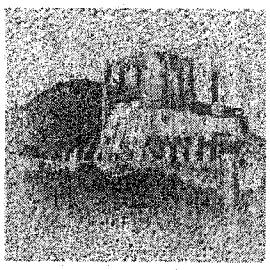
Excavator Cost ($/hr) : **$398.00**


List of Support Equipment : Form
Support Equipment Reports
Track Excavators (Support)
Motor Graders (Support)
Soil Compactors (Support)
Track Tractors (Support)
Water Tankers (Support)
Close

Support Equipment Report

Soil Compactors

Compactor Model : 825G



Compactor Characteristics:

Flywheel (KW)	:	235
Engine Model	:	3406C DITA
Rated Engine (RPM)	:	2100
No. of Cylinders	:	6
Feet Per Wheel	:	65
Two Pass Width (m)	:	4.88

Blade Specifications:

BladeType	:	Fill Spreading
Blade Capacity (CM)	:	3.79
Blade Weight (Ton)	:	2.83
Blade Width (m)	:	8.37
Blade Height (m)	:	4.61
Operating Weight (Ton)	:	31.7
Tank Capacity (L)	:	630
Compactor Cost ($/hr)	:	$90.00

Hourly Owning & Operation Costs

Machine Designation	MODEL
FRONT SHOVELS	5130B

OWNING COSTS

Depreciation	176.40
Interest	50.45
Insurance	5.05
Taxes	5.05
	$ 236.94

OPERATING COSTS

Fuel	59.40
Lube Oils, Filters, Grease	6.73
Tires	0.00
Undercarriage	12.38
Repair Reserve	21.00
Special Wear Items	25.00
	$ 124.51

OPERATOR'S HOURLY WAGE	$ 37.50
TOTAL COST	$ 398.95

GA PARAMETERS

Population Data

Senario Population Size (Even) :	20
No. of Generations :	3

Probabilities

Crossover :	0.70
Mutation :	0.05
No of Simulation Runs:	5
Extreme Configrations:	Allowed
User Defined Configration(s):	Allowed
Elitism:	Allowed
Non-Improvement Stop:	Allowed

GENERATIONS AND THEIR FITNESSES

Total No. of Generations: 20

No. of Scenarios: 3

Gen. No.	Scen. No.	Pile Equipme	Loaders	Haulers_1	Haulers_2	Spreaders	Compactores	Fitness
1	1	---	2	10	---	1	1	24,375,500
	1	---	8	60	---	8	8	18,898,500
	1	---	4	30	---	3	3	21,144,200
	1	---	2	35	---	6	8	41,792,000
	1	---	3	19	---	5	4	19,648,800
	1	---	6	51	---	3	6	22,615,300
	1	---	7	47	---	1	3	26,783,300
	1	---	2	43	---	4	7	46,430,900
	1	---	5	45	---	2	2	25,390,200
	1	---	4	22	---	3	7	17,775,500
	1	---	7	22	---	7	7	12,627,400
	1	---	7	28	---	6	7	13,594,100
	1	---	4	36	---	4	2	26,197,700
	1	---	3	23	---	1	3	28,273,900
	1	---	3	28	---	6	6	25,093,900
	1	---	4	10	---	5	7	15,744,600
	1	---	6	33	---	7	5	16,819,400
	1	---	4	40	---	6	5	24,527,900
	1	---	7	36	---	5	1	27,795,900
	1	---	5	42	---	4	6	21,592,500
	2	---	2	10	---	1	1	24,613,400
	2	---	8	60	---	8	8	19,680,900
	2	---	3	35	---	3	4	29,020,000
	2	---	8	26	---	5	8	14,399,500
	2	---	3	43	---	4	2	35,920,700
	2	---	5	27	---	4	8	17,018,400
	2	---	2	15	---	6	1	36,042,000
	2	---	7	60	---	7	3	24,498,700
	2	---	4	48	---	4	5	28,566,800
	2	---	2	36	---	5	2	45,098,100
	2	---	8	18	---	5	2	20,541,200
	2	---	6	37	---	2	6	20,885,100
	2	---	3	18	---	3	7	19,825,400
	2	---	3	12	---	2	6	17,328,600
	2	---	3	40	---	7	5	35,059,100
	2	---	3	27	---	3	8	24,676,300
	2	---	2	20	---	3	4	29,661,900
	2	---	4	23	---	7	8	20,240,900
	2	---	5	30	---	1	2	25,865,900
	2	---	8	22	---	3	7	16,290,100
	3	---	2	10	---	1	1	26,538,900
	3	---	8	60	---	8	8	24,361,100

	3	---	4	40	---	3	4	30,683,500
	3	---	4	35	---	6	1	40,060,900
	3	---	3	60	---	7	1	65,231,800
	3	---	7	50	---	5	6	22,941,800
	3	---	7	43	---	8	5	22,545,600
	3	---	3	11	---	2	4	21,145,800
	3	---	7	49	---	4	6	22,232,100
	3	---	7	35	---	6	5	19,199,200
	3	---	3	46	---	5	2	46,128,500
	3	---	6	17	---	5	7	21,193,200
	3	---	8	52	---	6	5	21,499,400
	3	---	6	45	---	2	7	25,882,400
	3	---	6	25	---	1	4	24,503,400
	3	---	6	57	---	2	7	29,877,300
	3	---	6	27	---	7	1	31,681,300
	3	---	7	24	---	4	7	17,501,200
	3	---	3	27	---	6	6	34,671,200
	3	---	6	17	---	7	6	22,888,700
2	1	---	4	40	---	6	6	24,500,000
	1	---	3	28	---	7	5	25,779,300
	1	---	3	28	---	6	8	25,119,400
	1	---	8	17	---	8	6	14,813,500
	1	---	3	23	---	1		
	1	---	8	60				
								19,606,300
					---	4	7	17,501,200
	3	---	7	26	---	4	6	16,877,100
	3	---	7	36	---	7	6	20,006,400
	3	---	7	24	---	4	6	17,489,900
20	1	---	4	40	---	6	6	24,500,000
	1	---	4	40	---	6	6	24,500,000
	1	---	4	28	---	6	3	21,576,900
	1	---	4	40	---	6	6	24,500,000
	1	---	4	40	---	6	6	24,500,000
	1	---	4	29	---	6	6	20,074,400
	1	---	4	40	---	6	3	26,425,300
	1	---	4	40	---	4	6	24,167,500
	1	---	4	28	---	6	6	19,606,300
	1	---	4	40	---	6	6	24,500,000
	1	---	3	41	---	6	3	32,928,500
	1	---	4	40	---	6	6	24,500,000
	1	---	4	40	---	6	5	24,527,900
	1	---	4	28	---	6	6	19,606,300
	1	---	5	40	---	6	3	22,797,200
	1	---	4	40	---	6	6	24,500,000
	1	---	7	22	---	7	7	12,627,400
	1	---	4	58	---	6	3	33,721,300
	1	---	4	40	---	6	6	24,500,000
	1	---	4	40	---	6	6	24,500,000

APPENDIX C

EARTHMOVING MARKUP APPLICATION (EMMA)

C.1 General

This appendix describes the essential steps followed in the development and implementation of EarthMoving Markup Application (*EMMA*). It also presents a numerical example, designed to illustrate its use.

C.2 Construct Decision Hierarchy

All attributes that influence bidding decisions and markup estimating should be identified by the decision-maker in order to construct the decision hierarchy that best suits his/her bidding strategy. Attributes that have common characteristics can be clustered into a group that forms the evaluation criteria in the decision hierarchy. The user is then required to define all information necessary to construct the decision hierarchy: 1) total number of hierarchy levels; 2) total number of evaluation criteria; 3) total number of attributes, 4) number of intermediate criteria (IC) that end with sub-criteria, 5) maximum number of sub-criteria that belong to the same IC, 6) number of criteria (CA) that end with attributes and 7) maximum number of attributes, associated with any criteria. Information regarding each criterion and attribute should also be entered. Data entry can be carried out either from a previously stored external MS Excel file, as shown in Figure C.1, or through the interactive dialog boxes shown in Figure C.2.

C.3 Define Relative Importance

A procedure similar to the AHP (Saaty 1982) has been used to generate the

relative weights associated with the attributes considered in the decision hierarchy. The procedure is based on pairwise comparison among the attributes using a numerical scale from 1 to 9, where 1 indicates equal importance of the two attributes under consideration and 9 indicates absolute importance of one attribute over the other (see Table C.1).

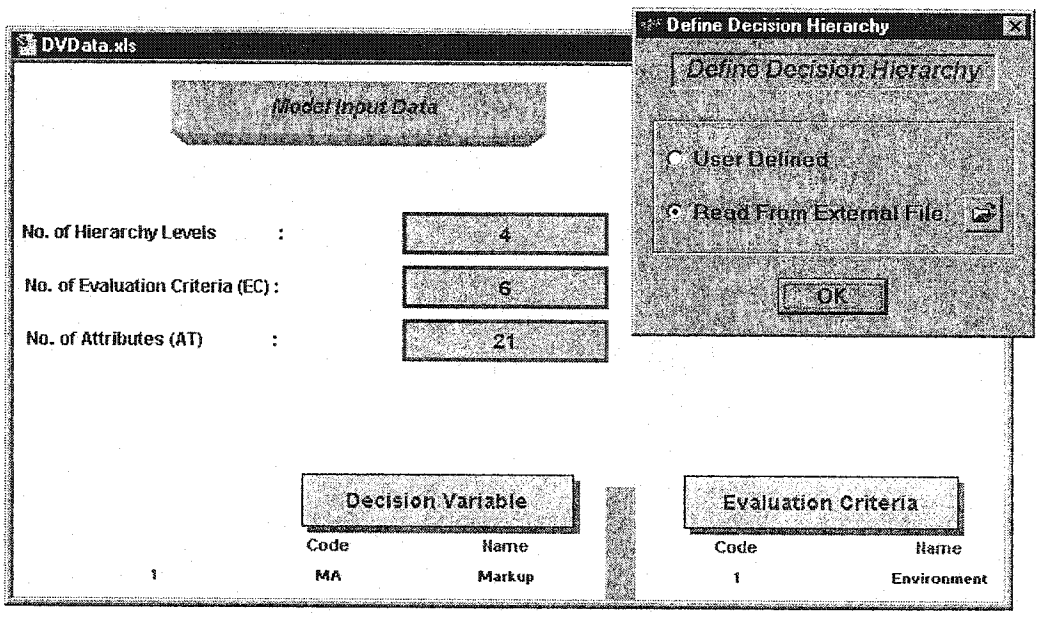


Figure C.1 Constructing Decision Hierarchy Using External File

Table C.1 Pairwise Comparison Scale (Saaty 1982)

Relative Importance	Definition
1	Equal importance of both attributes
3	Weak importance of one attribute over another
5	Essential or strong importance of one attribute over another
7	Demonstrated importance of one attribute over another
9	Absolute importance of one attribute over another
2, 4, 6, 8	Intermediate values between two adjacent judgements

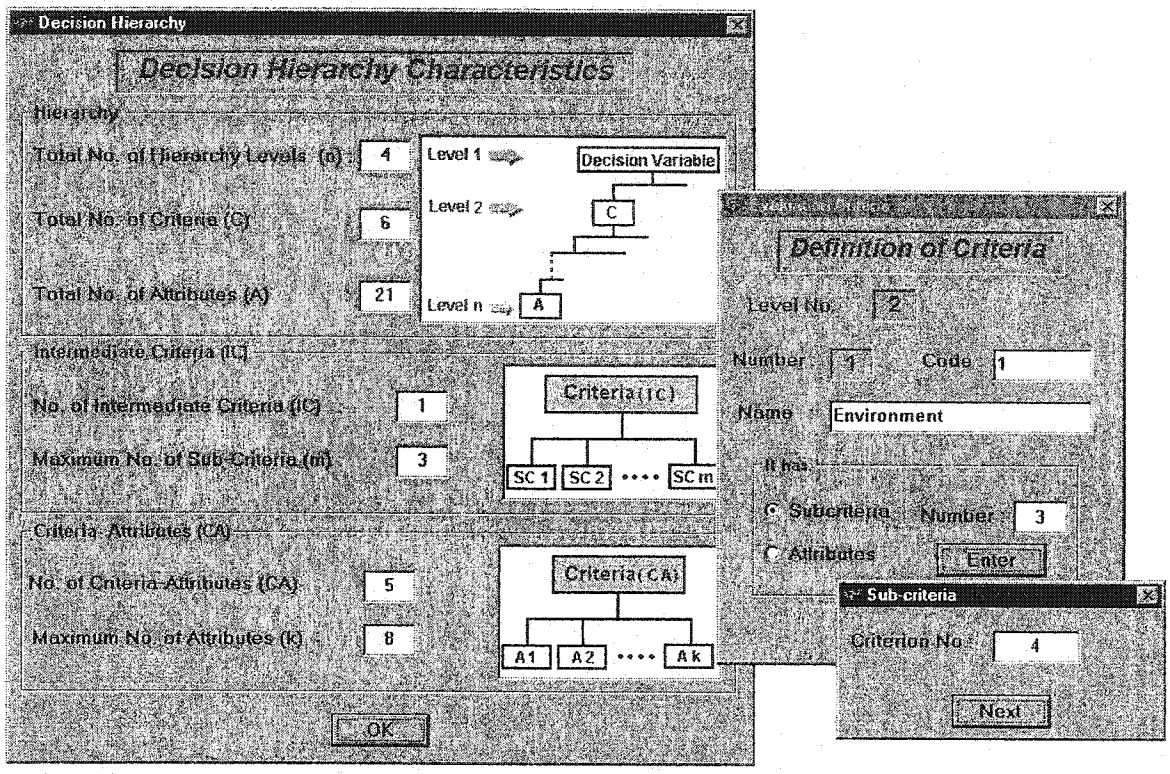


Figure C.2 Constructing Decision Hierarchy Using Dialog Boxes

Upon completion of the pairwise comparisons and formation of the matrix of comparisons, the eigenvalues and eigenvectors are then obtained. The eigenvector represents directly the relative weights among the attributes (i.e. their relative importance) and the eigenvalue represents the consistency of the decision-maker in assigning the relative importance among the attributes during the process of pairwise comparisons. The developed procedure has been coded using MS Visual Basic 6.0.

Unlike the models developed for estimating markup percent (Seydel and Olson 1990, Dozzi et al 1996, Chua and Li 2000), the proposed *EMMA* is neither limited

to a fixed number of criteria and attributes nor requires the use of external software such as in the case of Chua and Li (2000). A dialog box has been designed (see Figure C.3) to facilitate the process of pairwise comparisons and the display of the resulting relative weights (among the evaluation criteria, and attributes). The same dialog box could be used to revise the data entered, in an effort to obtain a satisfactory consistency. Generally, a consistency ratio of less than or equal to 0.1 is considered acceptable (Saaty 1982).

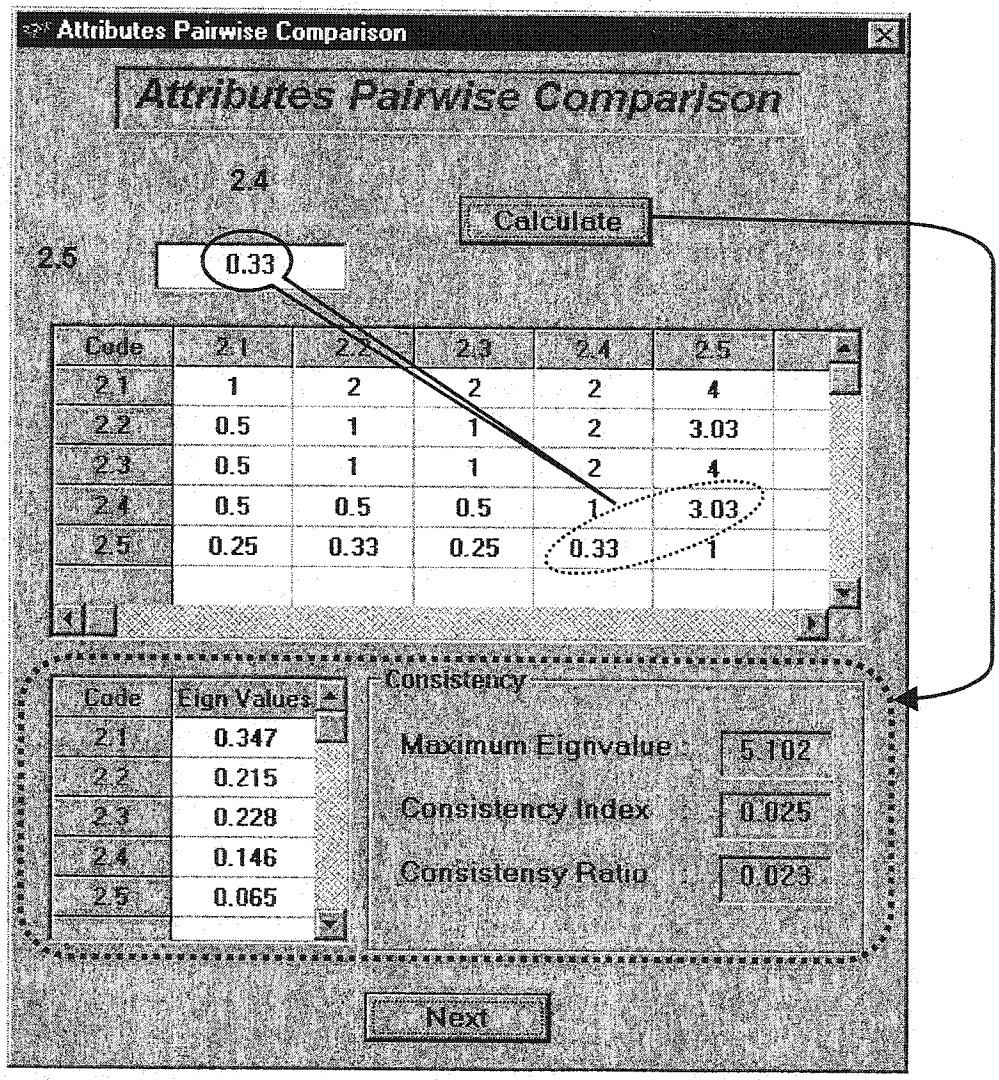


Figure C.3 Pairwise Comparison Dialog Box

C.4 Select Utility Functions

Utility functions are used to quantify the preference of the decision-maker by assigning a numerical index to various degrees of satisfaction as the attribute under consideration takes values between the most and least desirable limits. These limits are defined for each attribute using any preferred units such as percentage, months, dollars ($), etc. (see Figure C.4). It has been recommended that utility functions be monotonic (Keeney and Raiffa 1993) in such a way that the most desirable scenario corresponds to the highest utility [$u(x_i)$=1.0], whereas the least desirable scenario corresponds to the lowest utility [$u(x_i)$=0].

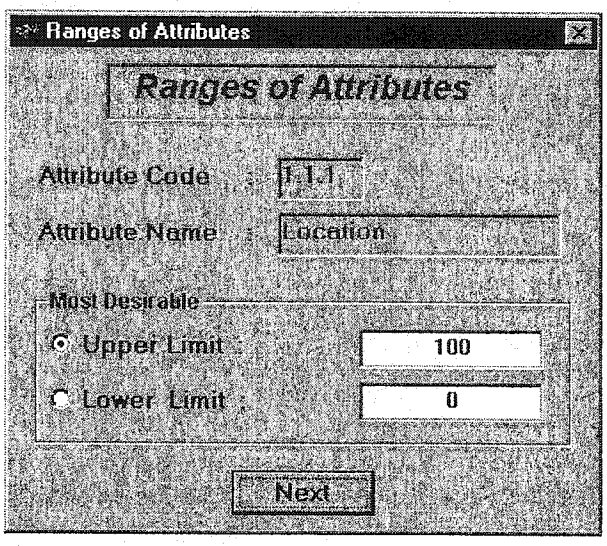


Figure C.4 Defining Ranges of Attributes

Decision-makers' attitude towards risk has been accounted for using a set of utility functions. In the *Risk Averse* attitude, the value of certainty equivalent, $\hat{x}$, (that corresponds to 0.5 utility) is less than the average value of the attributes'

limits (x_L and x_U). In this case the condition ($\hat{x} < \frac{x_L + x_U}{2}$) must be satisfied. The utility function that represents this attitude can be expressed either by a piece-wise linear function (Equations C.1-a, C.1-b and C.2; and Figure C.5) or an exponential function (Equation C.3; Figure C.6)

$$u_1(x) = ax \qquad \text{(C.1-a)}$$

$$u_2(x) = bx + c \qquad \text{(C.1-b)}$$

$$u(x) = \frac{u_1(x) + u_2(x)}{2} \qquad \text{(C.2)}$$

$$u(x) = ae^{bx} + c \qquad \text{(C.3)}$$

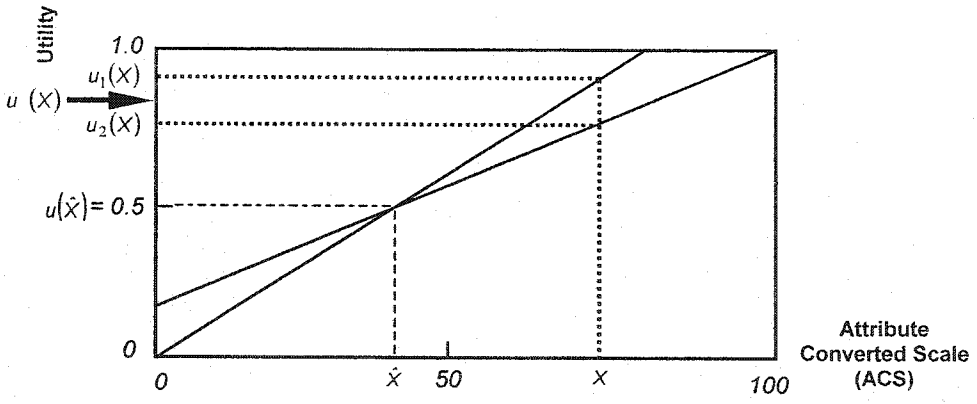


Figure C.5 Piece-wise Linear Utility Function (Risk Averse Attitude)

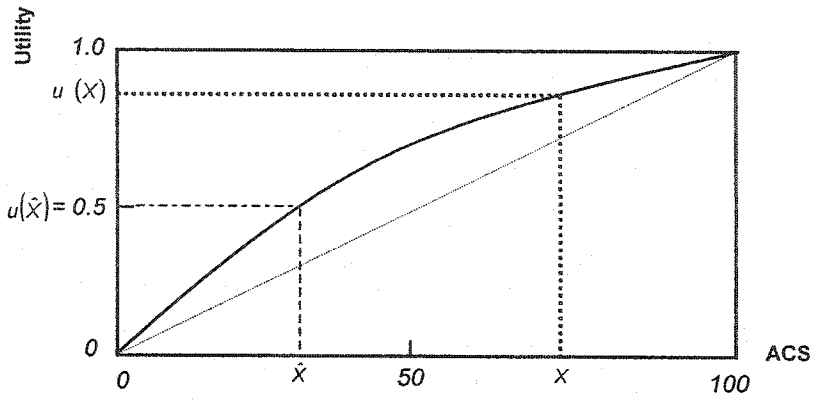


Figure C.6 Exponential Utility Function (Risk Averse Attitude)

The utility functions in the *Risk prone* attitude case can, similarly, be expressed either by a piece-wise linear function (Equations C.1-a, C.1-b and C.2; and Figure C.5) or a logarithmic function (Equation C.4; Figure C.7)

$$u(x) = a\ln(x+b)+c \qquad \text{(C.4)}$$

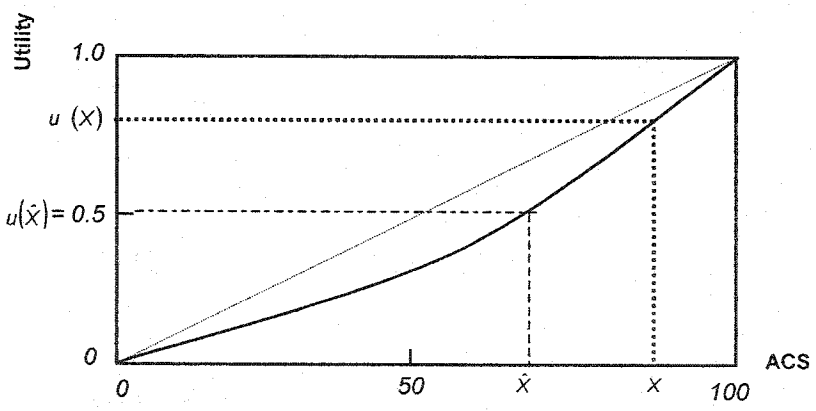


Figure C.7 Logarithmic Utility Function (Risk Prone Attitude)

In the *Risk neutral* attitude, the value of certainty equivalent is equal to the average of the attributes' limits ($\hat{x} = \frac{x_L + x_U}{2}$). The utility function in this case can be expressed by a straight line (Equation C.5).

$$u(x) = ax \qquad \text{(C.5)}$$

It should be noted that the coefficients *a, b* and *c*, in the above equations, are determined by satisfying the conditions at: 1) lowest utility, 2) certainty equivalent and 3) highest utility. These functions have been coded in the proposed model. A specially designed dialog box has been developed to facilitate data entry (see Figure C.8).

C.5 Expected Utility Value

Upon constructing the decision hierarchy and selecting the appropriate utility functions, the user is required to enter for each attribute: 1) the attribute's value at certainty equivalent; and 2) the value of that attribute in the project being considered. Figure C.8 depicts the dialog box designed to enter such values. Subsequently, the model calculates the weight associated with each attribute. For example, the weights associated with location, labor reliability, and labor availability can be calculated as shown in Table C.2.

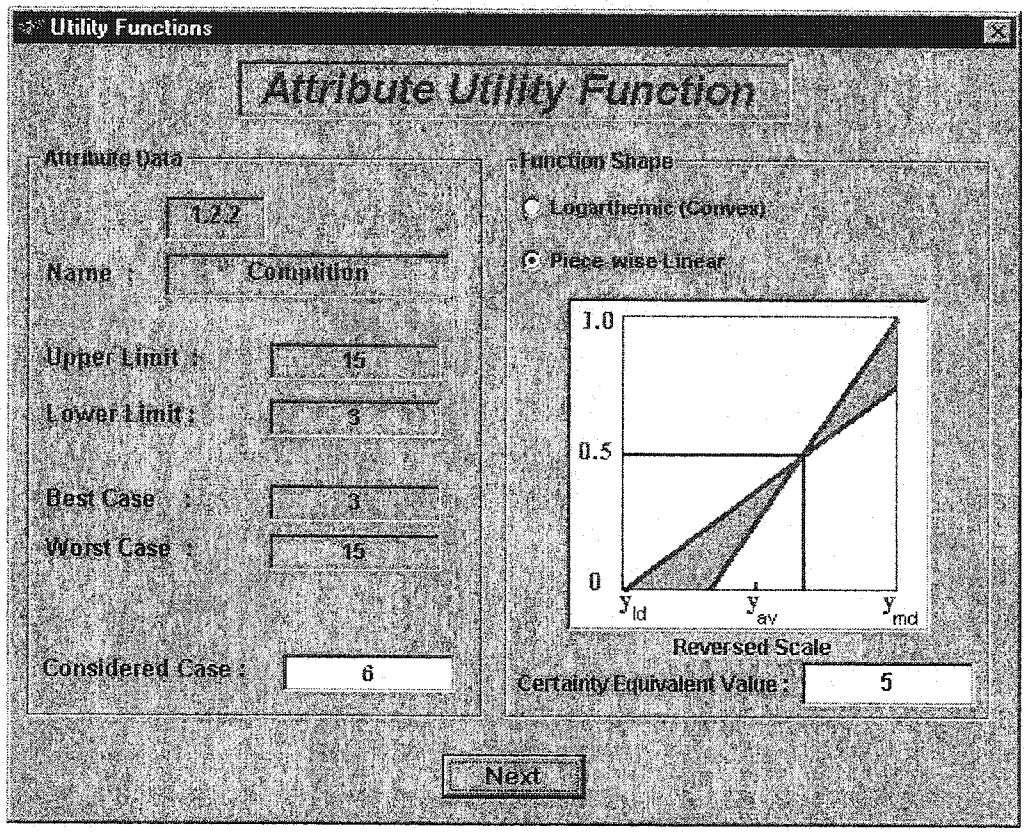


Figure C.8 Attribute Utility Function Dialog Box

Table C.2 Eigenvectors Results

Attribute / Criterion	Criteria Weight	Attribute Weight
1 External Factors	0.285	---
1.1 Geographic Factors	0.31	---
1.1.1 Location	0.4	0.035
1.1.2 Labor Reliability	0.2	0.018
1.1.3 Labor availability	0.4	0.03

Assuming mutually independent attributes, a simple additive utility model, is used (Keeney and Raiffa 1993):

$$u(x_1, x_2, \cdots, x_n) = \sum_{i=1}^{n} k_i u_i(x_i) \qquad \text{(C.6)}$$

In which k_i: positive weight for attribute i; and u_i: utility value for attribute i. As a result, the total weighted utility (TWU) is equal to the sum of expected utilities of all attributes in the decision hierarchy (see Table C.3).

Table C.3 Expected Utility Value

Attribute	Utility Value	Weight	Expected Utility
1.1.1 Location	1	0.04	0.04
1.1.2 Labor reliability	0.6	0.02	0.012
.	.	.	.
.	.	.	.
3.8 Estimate uncertainty	0.03	0.06	0.0018
Total Weighted Utility (TWU)			**0.687**

C.6 Markup Utility Function

The characteristics of the markup utility function should be defined in a similar manner to that used for each attribute by defining: 1) largest value, 2) lowest value and 3) certainty equivalent. The estimated total weighted utility (TWU) is used to calculate the recommended markup percentage as shown in Figure C.9.

Figure C.10 depicts a dialog box designed for defining the characteristics of the markup function.

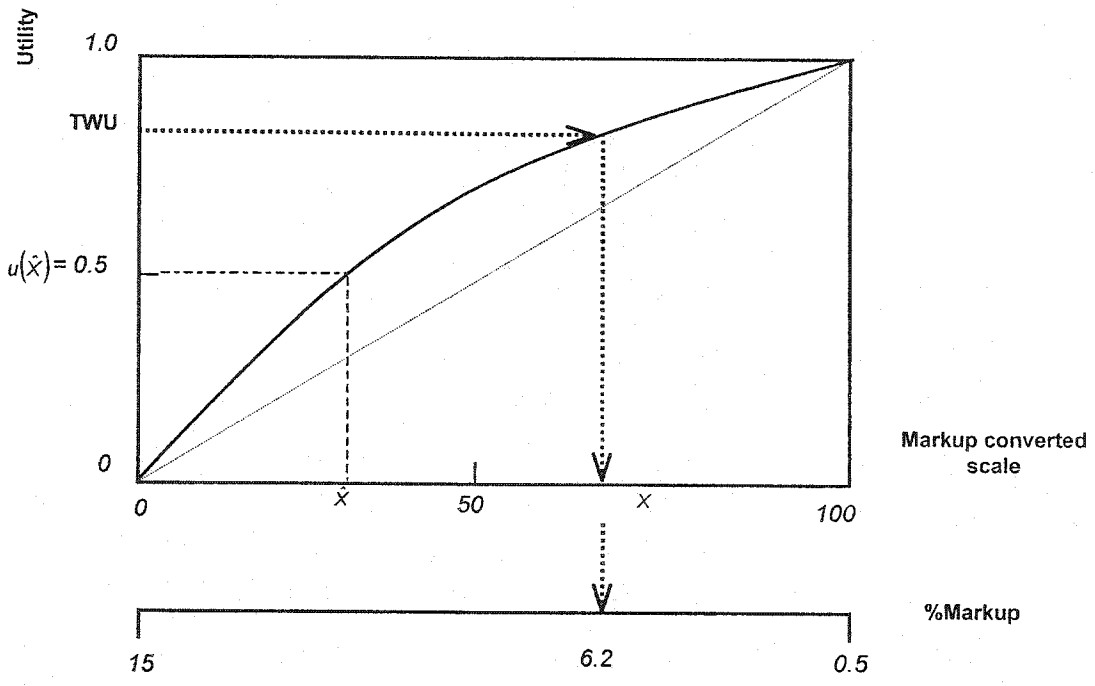


Figure C.9 Recommended Markup Percent (Exponential Utility Function)

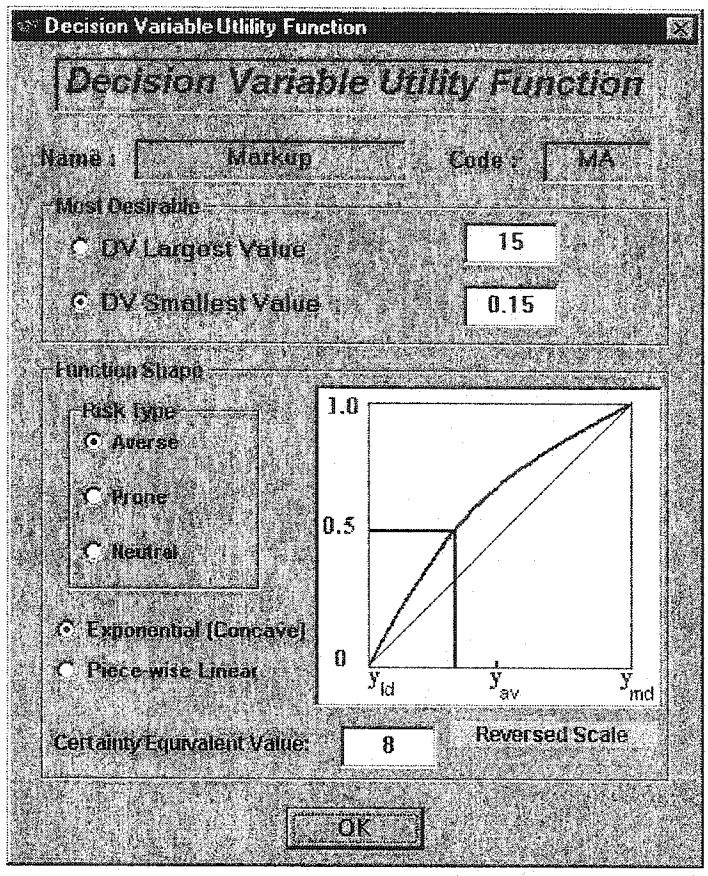


Figure C.10 Markup Utility Function Dialog Box

C.7 Numerical Example

A numerical example from the literature is analyzed to validate and demonstrate the different features of the developed *EMMA*. This example considers the project presented by Dozzi et al (1996) for estimating markup percent. The decision hierarchy used for that project consists of 21 attributes adopted from Ahmed and Minkarah (1988-b). These attributes have been grouped into three criteria: 1) environmental, 2) company and 3) project factors (see Figure C.11).

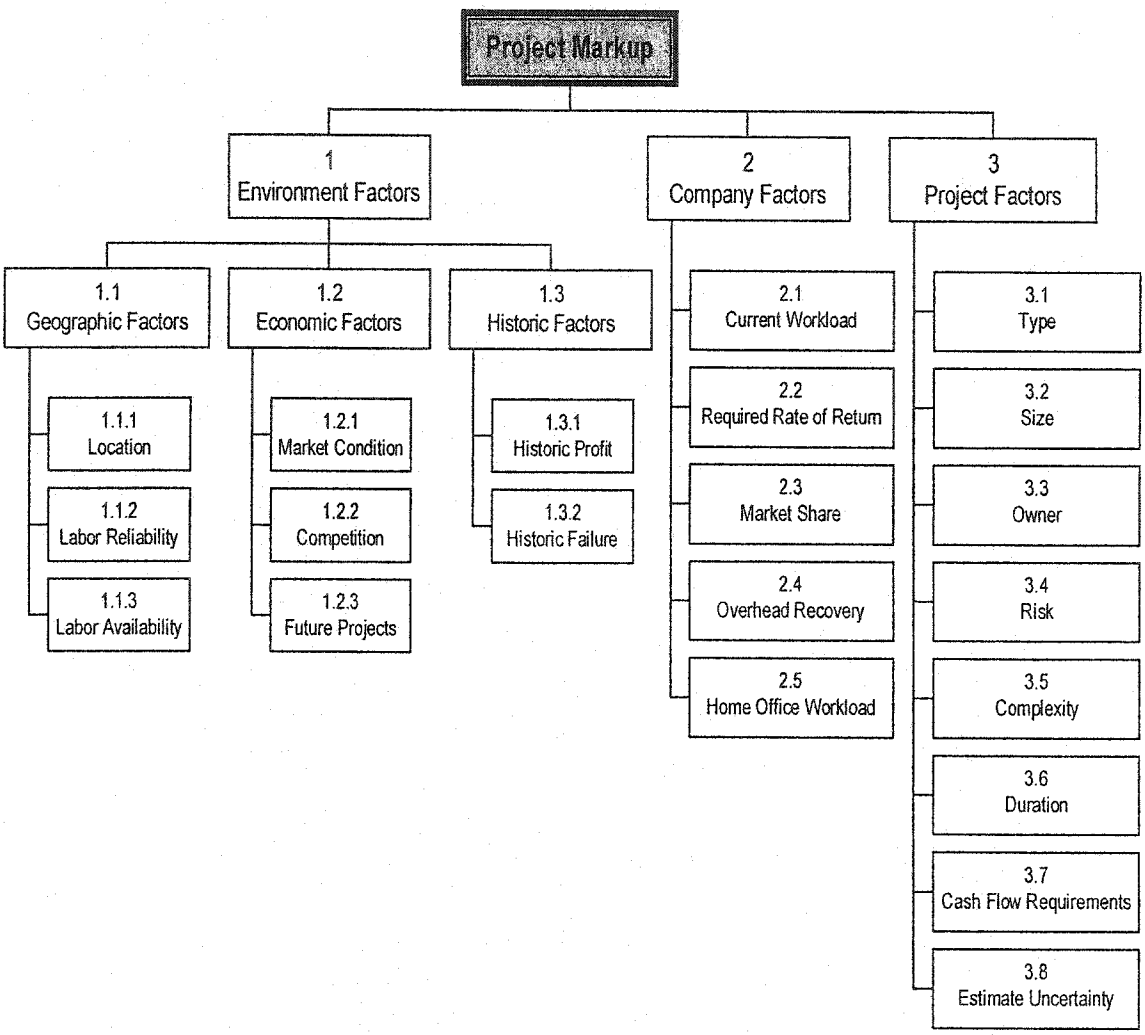


Figure C.11 Markup Decision Hierarchy (Dozzi et al 1996)

The percent markup calculated using the proposed model is 2.53% (see Figure C.12), compared to the 3.94% calculated by Dozzi et al (1996). It should be noted that the difference can be attributed to their approximation of piece-wise linear utility functions. Figure C.13 depicts the *EMMA*'s reports menu. The *EMMA*'s generated reports are shown in Figures C.14, C.15, C.16, C.17 and C.18.

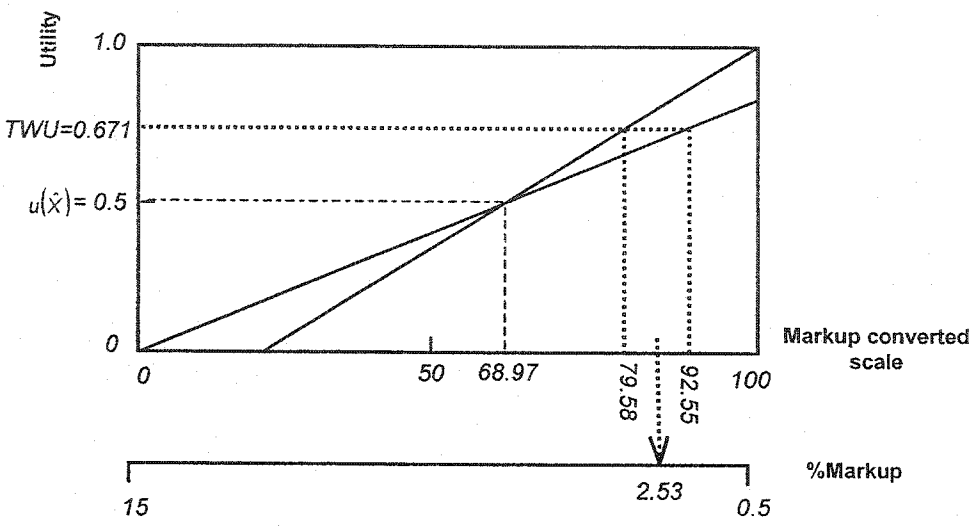


Figure C.12 Calculated Markup

In order to demonstrate the impact of the shape of utility functions on the estimated markup percent, different shapes are assigned to the markup utility function to represent different scenarios of risk attitudes. The results are summarized in Table C.4. One can also examine the impact of the functions associated with the model's attributes.

Table C.4 Different Shapes of Markup Utility Functions

	Scenario 1 (Risk Prone)	Scenario 2 (Risk Prone)	Scenario 3 (Risk Neutral)	Scenario 4 (Risk Averse)	Scenario 5 (Risk Averse)
Utility Function Type	P-wise	Exponential	Straight line	P-wise	Logarithmic
Certainty Equivalent	5	5	7.75	10	10
Constant (a)	0.00725	33.13128	0.01	92.55022	0.0145
Constant (b)	0.01611	3798	0	0.00016	0.007632
Constant (c)	-0.61111	-273.0756	0	-92.55022	0.23684
Markup (%)	2.53	3.74	5.28	8.3	7.53

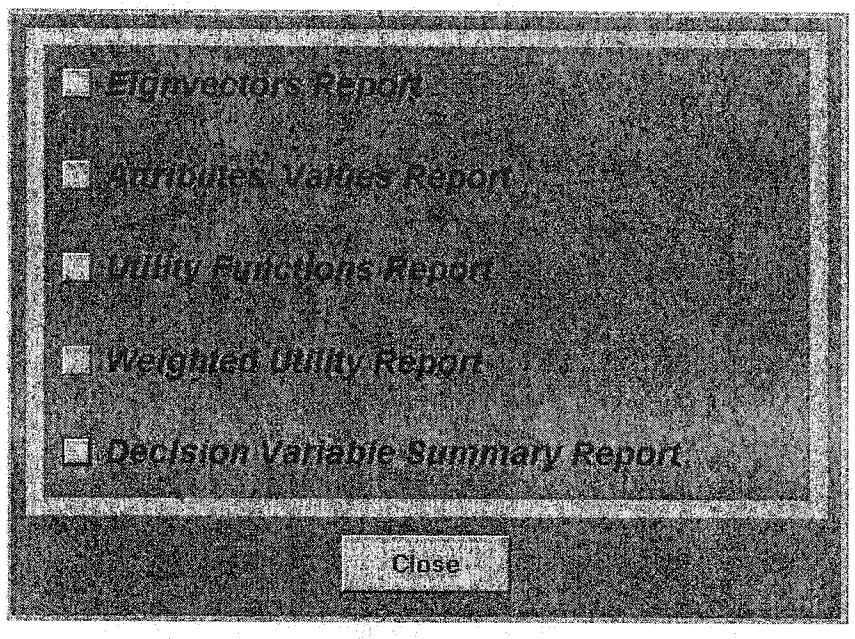


Figure C.13 EMMA's Reports Menu

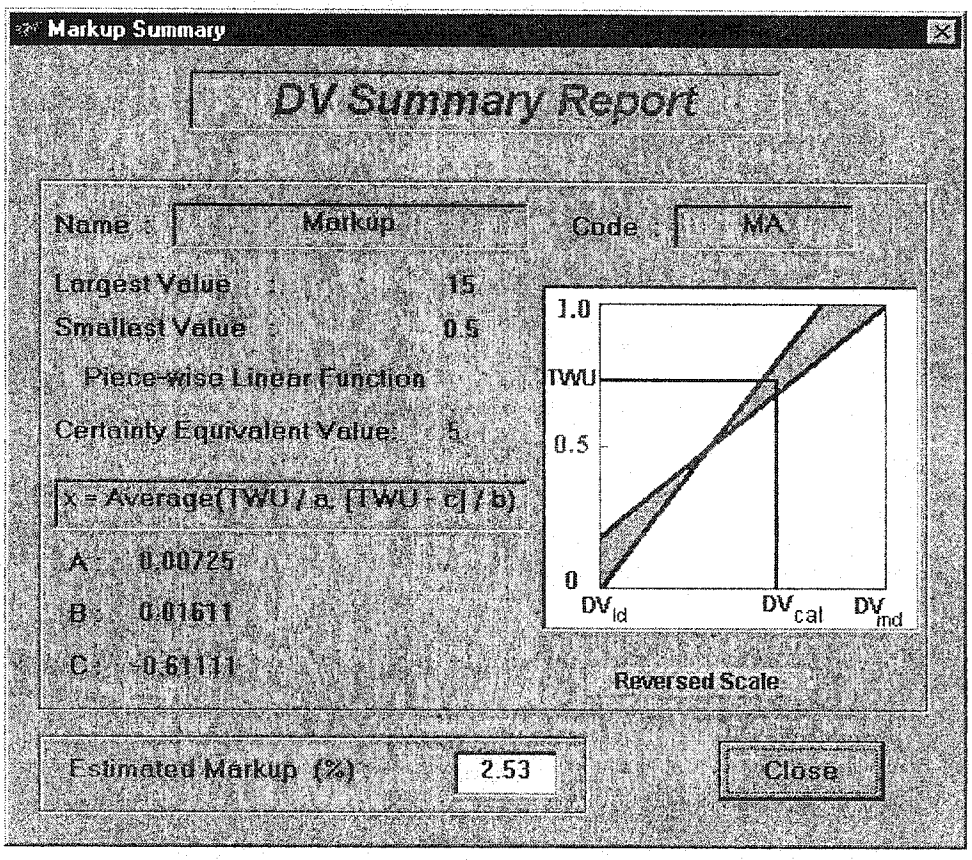


Figure C.14 Markup Summary Report

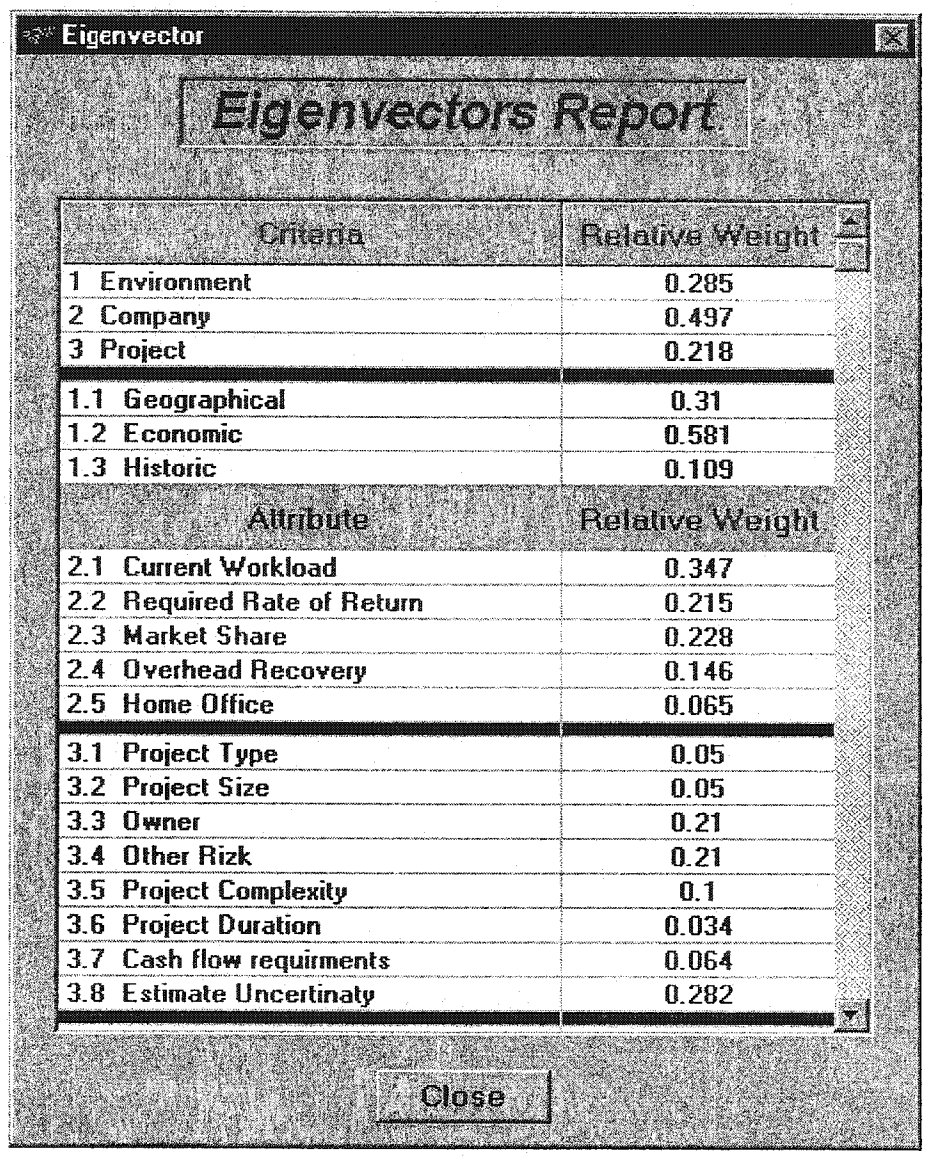


Figure C.15 Eigenvector Report

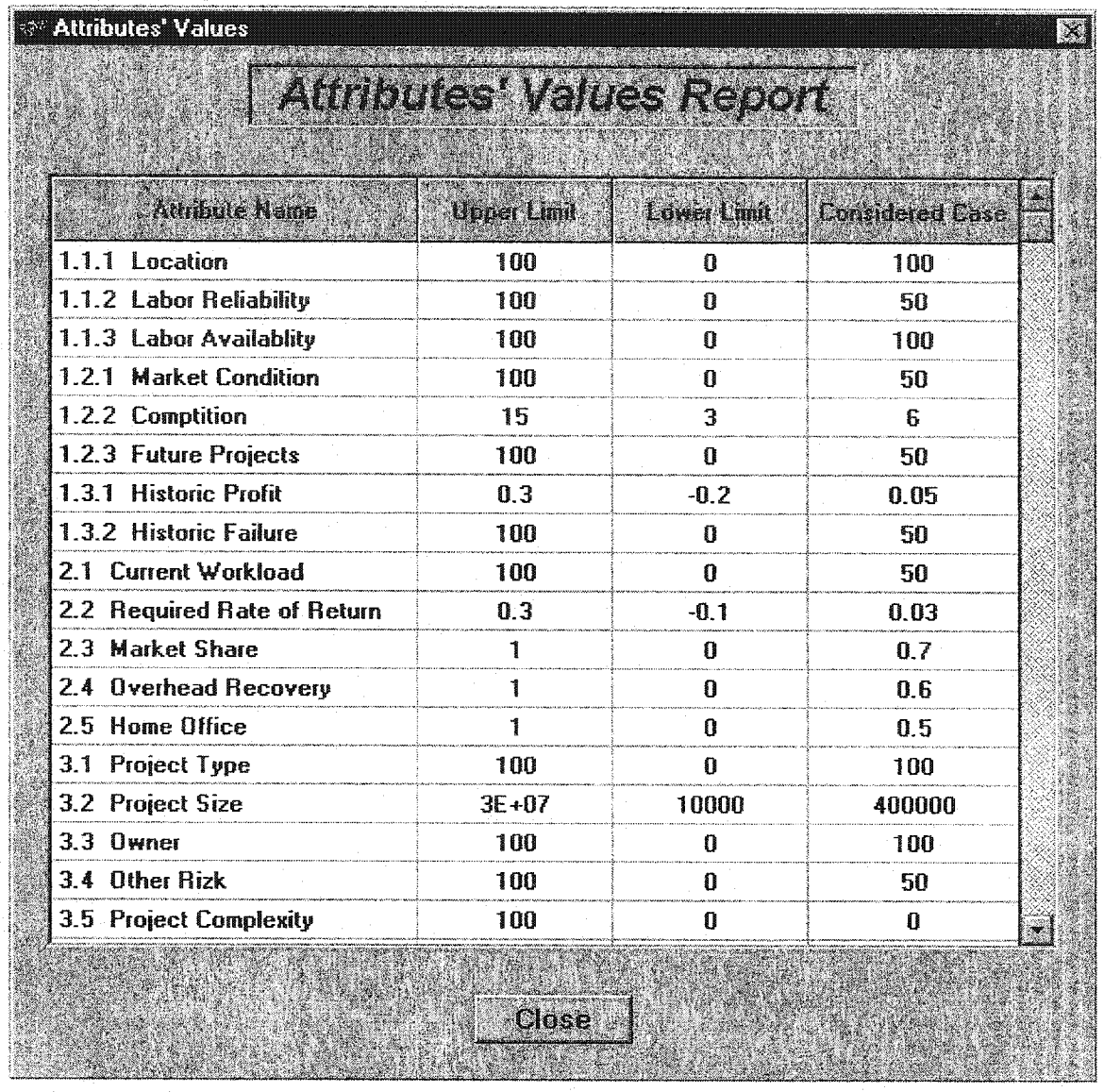


Figure C.16 Attributes' Value Report

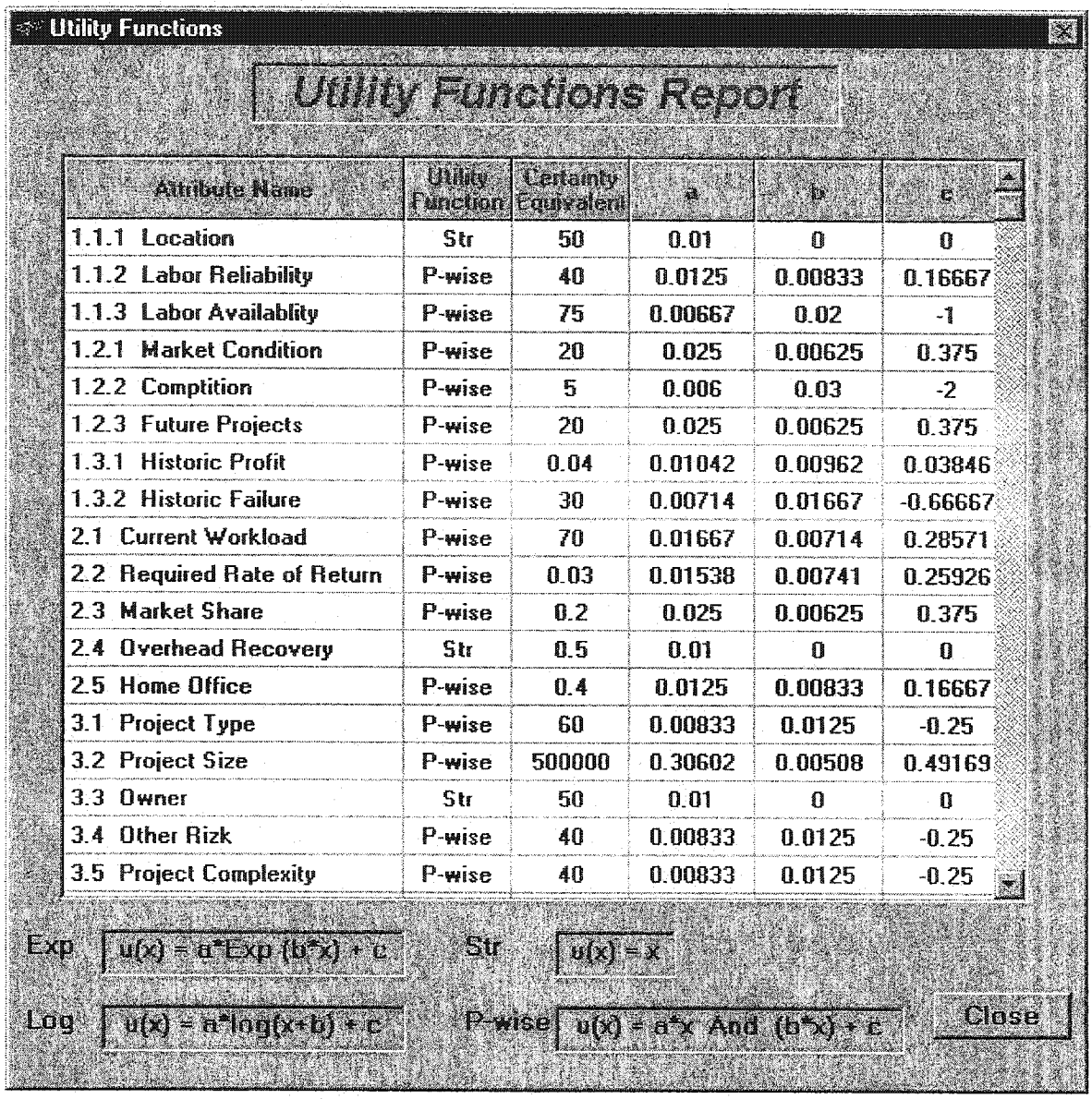


Figure C.17 Utility Function Report

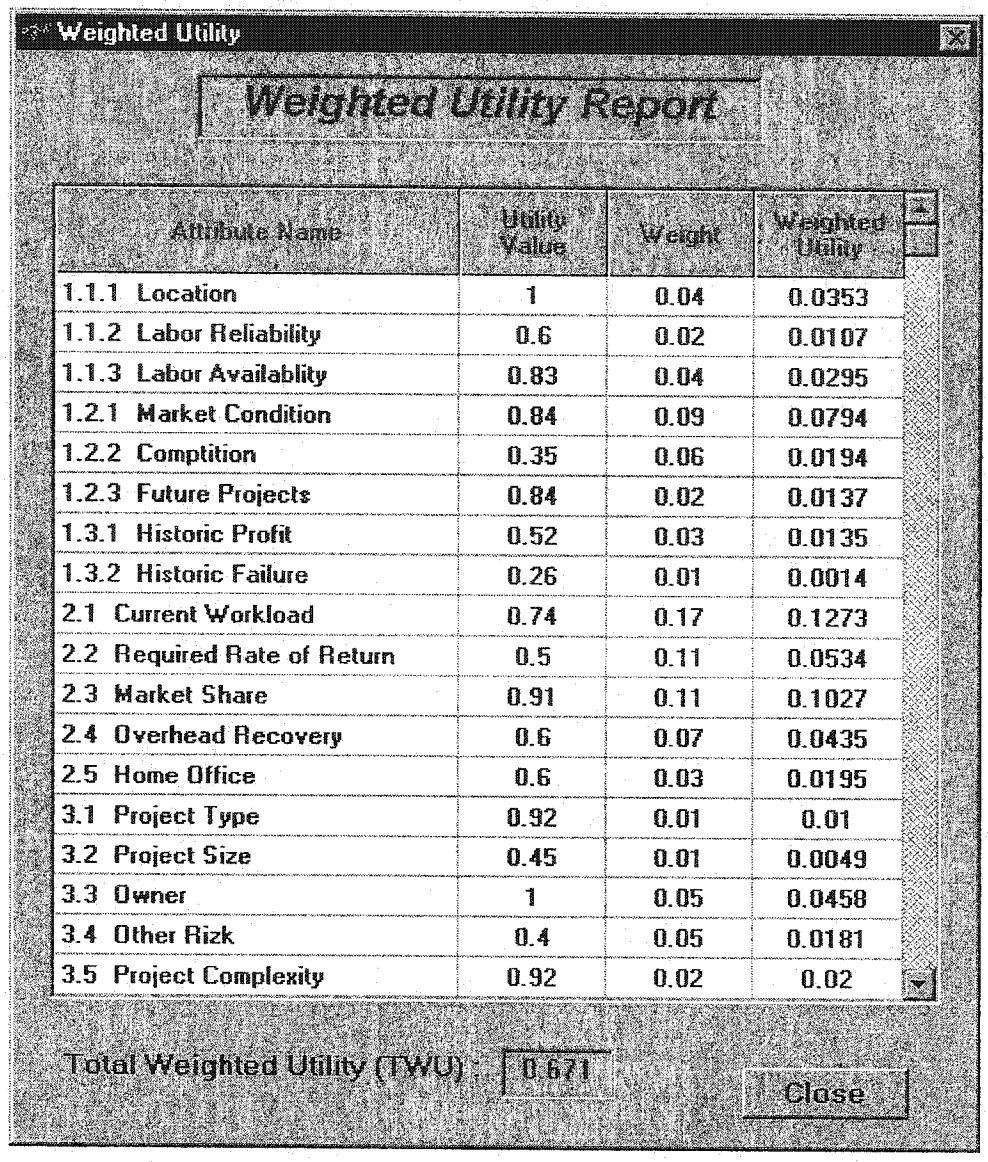


Figure C.18 Weighted Utility Report